

Module 6

Real-World Application



Table of Contents

Module 6: Real World Application

Overview

Introducing Liferay Service Builder

Exercise: Create the Assignment Service

Exercise: Implement Assignment Local Service

Exercise: Implement Assignment Remote Service

Create the Presentation Layer

Exercise: Create the Gradebook Web Module

Exercise: Implement the Main View

Exercise: Implement the Assignment Editing View

Exercise: Implement Validation

Exercise: Add Localization Resources

Exercise: Add CSS Resources

Implement Access Control

Exercise: Implement Service Module Permissions

Exercise: Implement Web Module Permissions

Integrate with Liferay Frameworks

Exercise: Integrate with the Asset Framework

Exercise: Integrate with Portal Search

Make the Application Configurable

Implement Workflow Support

Integrate with External Systems

Logging

Testing

Debugging

Managing Deployment Issues

Optional Exercises

Optional Exercise: Test the Service Through Remote API

Optional Exercise: Implement Submissions

Optional Exercise: Add JavaScript Resources

Optional Exercise: Make the Gradebook Configurable

Optional Exercise: Enable Workflows for Assignments

Optional Exercise: Publish a REST Service

Optional Exercise: Implement Gradebook Logging

Optional Exercise: Implement Integration Tests

Optional Exercise: Debug the Gradebook

Optional Exercise: Troubleshoot Module Deployment

Module 6: Real World Application

Learning Objectives

In this module, you'll understand how to build a complete real-world liferay application, understand how to connect your business logic to your user interface, and learn how to leverage time-saving liferay tools and frameworks in a single application.

Tasks to Accomplish

- Define the Assignment entity and create the service layer for the Gradebook application using the Liferay Service Builder code generation tool
- Use the local service implementation class to add custom logic to the Assignment local service
- Implement the remote service variant and deploy the service to the Liferay server
- Create the user interface for the Gradebook application
- Implement a master-detail view for assignments
- Implement the assignment editing view and the related MVC Action Commands
- Implement Assignment validation on both the service and user interface layers and provide feedback to the user interface
- Add and Implement localization resources
- Add and configure CSS resources
- Implement permission support and checking on the service layer
- Implement permissions support and checking on the user interface layer
- Integrate the Gradebook application with Liferay's Asset Framework
- Integrate the Gradebook application with the Liferay Search framework

Exercise Prerequisites

- Java JDK Installed to Run Liferay
 - Download here:
<https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html>
 - Instructions on Installation here:
https://www.java.com/en/download/help/download_options.xml
- Unzipped module exercise files in the following folder structure:
 - Windows: `C:\liferay`
 - Unix Systems: `[user-home]/liferay`
- Liferay Developer Studio installed with a workspace selected
 - For installation instructions, see module 1
- Be prepared to use the code snippets found in the module's `exercise-src` folder
 - Code snippets for a particular exercise will be found in the folder with the corresponding exercise number: `exercise-[##]`
 - Snippets are named after the exercise set where they are used: `snippet-[mod#]-[exercise#]-[exercise-set-heading]-[optional-step#]`

Overview of Developing a Real World Application

In this chapter, we'll create a real-world Liferay application using Liferay's time-saving tools and frameworks.

In the required exercise steps, we'll build an assignment management system for a course gradebook. First, we'll define the Assignment entity and use the Liferay Service Builder code generation tool to create the database and service layer for it. Then, we'll create the user interface using the JSP technology and taking advantage of Liferay's tag libraries. Finally, we'll implement access control and integrate to Liferay's Asset and Search frameworks.

In the optional exercises we'll extend the application to support assignment submissions and grading, make the application configurable, enable workflows for the assignments, create a REST service, implement integration tests, and learn how to debug and manage deployment issues.

Features and requirements for the Gradebook application:

Application Features

- Teachers can create assignments.
- Students can send submissions to assignments.
- Teachers can grade the submissions.

Functional Requirements

- Assignments have to be listable in the **Asset Publisher** portlet
- Assignments have to be **searchable** with portal search
- Both assignments and submissions have to be under **access control**
- The Application has to be **configurable**
- The application has to support **localization**

Non-Functional Requirements

- The application has to be **modular**
- Data has to be **persisted** in the database

- Form submissions have to be **validated**
- There has to be basic level **integration tests**

Development Technologies

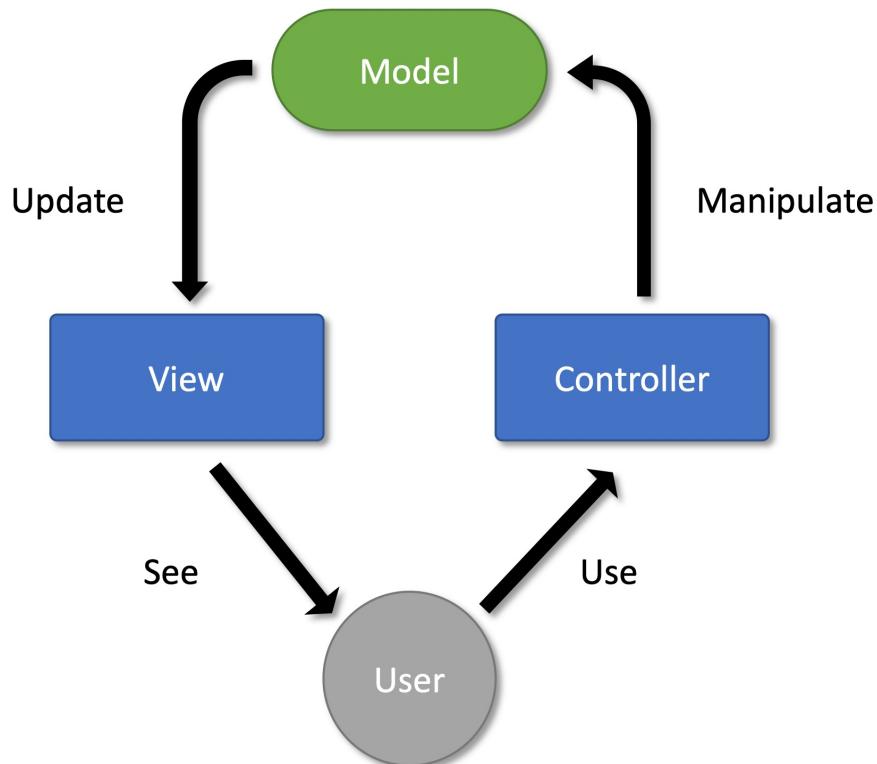
We will build the model and service layer of the Gradebook application using the *Liferay Service Builder* code generation tool, which greatly reduces the need for boilerplate coding, automatically creates the persistence layer, implements caching, and creates both local and remote service APIs.

Whether you'd prefer pure Java EE or OSGi, Liferay provides several options for front-end implementation. Blade portlet samples are available for many of them:

- <https://github.com/liferay/liferay-blade-samples/tree/7.2/liferay-workspace/apps>
- <https://github.com/liferay/liferay-blade-samples/tree/7.2/liferay-workspace/wars>

In our exercise application we will use the classic Liferay MVC portlet template with JSPs because of the many useful tag libraries available for this use case.

Like Liferay core applications, this application will follow the MVC design pattern:



Module Architecture

The application will be divided into four modules:

- **gradebook-api**: the service layer API
- **gradebook-service**: the service layer implementation
- **gradebook-web**: the user interface with portlet component

In the optional exercises, we will also create an integration test module called *gradebook-test*.

Implementation Steps Overview

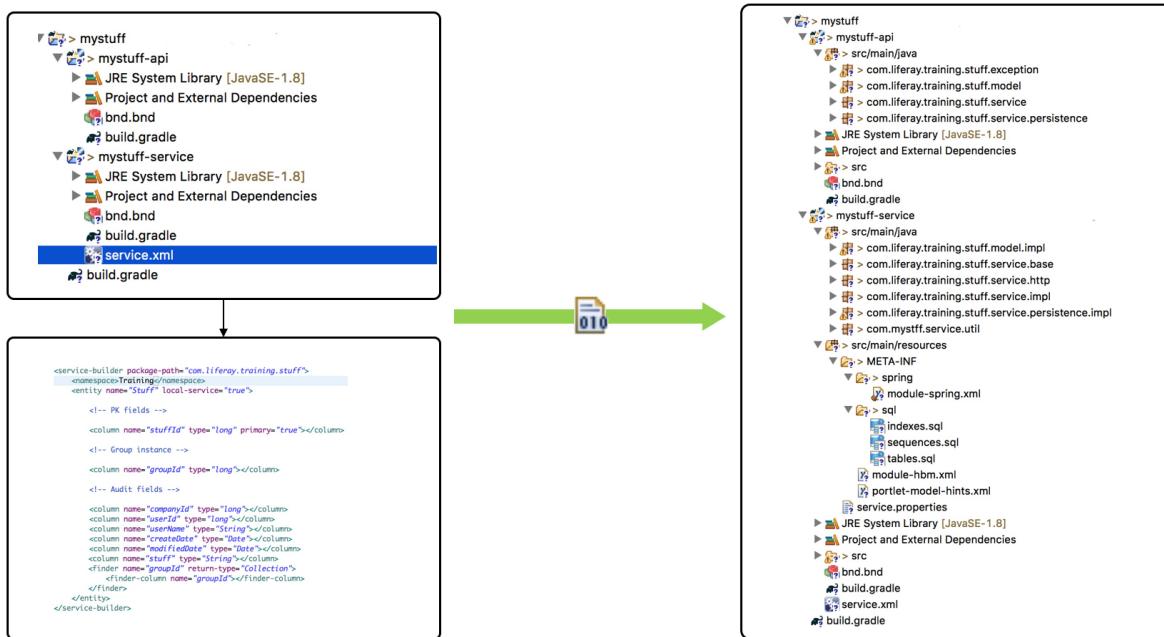
The main implementation steps are:

1. Create the API and service modules and define the data model
2. Create the portlet module and implement the user interface
3. Implement access control
4. Integrate with portal Asset and Search frameworks

Introduction

Liferay Service Builder is a code generation tool that takes an XML configuration file as an input and generates the complete service layer as an output. The generated code includes the database schema definition, persistence and caching code, service classes with CRUD methods, and a remote service layer supporting JSON and SOAP web services. The generated code can be complemented and overridden by service implementation classes, which are created for every entity defined in the service schema.

The diagram below illustrates a Service Builder project file structure before and after code generation:



A service created by the Service Builder defines a zone where all the operations are run within the same transaction. Service Builder can also be used just to create a web service, without defining any persistence entities.

Service Builder is one of the central development patterns in Liferay and is used in its core services. The persistence functionalities rely on [Hibernate](#), while in the implementation classes, you can choose whether to use [Spring](#) or, as of 7.2, OSGi Declarative Services. While generated service classes are Spring beans, they are wired to the OSGi service container and exposed through the OSGi service registry.

The generated database schema can be fine-tuned on a field-mapping level, and you can choose to use external datasources for any entity. Although Service Builder-generated code abstracts the database layer, providing basic CRUD methods, you can completely customize the service implementation classes and use dynamic and custom SQL queries in the code.

Note: You can only have a single datasource for a single Service Builder module. See the [Developer Network article for more information:
https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/connecting-service-builder-to-external-databases

Let's walk through the basic concepts of Liferay Service Builder.

service.xml

`service.xml` is the main Service Builder configuration file containing:

- Global information like the database namespace and package path for the service
- Entity definitions
- Default order of entity retrieval
- Entity finder methods
- Datasource configuration
- Custom service exception definitions
- Service references available in the generated service classes
- Caching information

Service class generation is done with the `buildService` Gradle task. Every time `service.xml` or signatures in the generated implementation classes are modified, the `buildService` task has to be re-run.

Below is an excerpt of the `service.xml` of [Liferay's Blogs application](#):

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder
7.2.0//EN" "http://www.liferay.com/dtd/liferay-service-builder_7
_2_0.dtd">

<service-builder auto-import-default-references="false" auto-name
space-tables="false" dependency-injector="ds" package-path="com.l
iferay.blogs">
```

```
<namespace>Blogs</namespace>
<entity local-service="true" name="BlogsEntry" remote-service="true" trash-enabled="true" uuid="true">

    <!-- PK fields -->

    <column name="entryId" primary="true" type="long" />

    <!-- Group instance -->

    <column name="groupId" type="long" />

    <!-- Audit fields -->

    <column name="companyId" type="long" />
    <column name="userId" type="long" />
    <column name="userName" type="String" uad-anonymize-field-name="fullName" />
    <column name="createDate" type="Date" />
    <column name="modifiedDate" type="Date" />
```



While we'll walk through the basic `service.xml` configuration in our Gradebook exercise, a lot of available options fall out of scope. See the [Service Builder DTD file](#) for all the options and documentation which you can later use in your own Service Builder projects.

When upgrading Service Builder projects to a newer portal versions, it's important to update the DTD to get all the improvements in the generated code within.

portlet-model-hints.xml

The `portlet-model-hints.xml` file is generated when the `buildService` task is run in `resources/META-INF`. It lets you customize the entity to SQL column mapping, like field types, column sizes, and validation.

Below is the `portlet-model-hints.xml` of [Liferay's Blogs application](#):

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

<model-hints>
    <model name="com.liferay.blogs.model.BlogsEntry">
        <field name="uuid" type="String" />
        <field name="entryId" type="long" />
        <field name="groupId" type="long" />
        <field name="companyId" type="long" />
        <field name="userId" type="long" />
        <field name="userName" type="String" />
        <field name="createDate" type="Date" />
        <field name="modifiedDate" type="Date" />
        <field name="title" type="String">
            <hint name="max-length">150</hint>
            <sanitize content-type="text/plain" modes="ALL" />
            <validator name="required" />
        </field>
        <field name="subtitle" type="String">
            <hint-collection name="TEXTAREA" />
        </field>
        <field name="urlTitle" type="String">
            <hint name="max-length">255</hint>
        </field>
        <field name="description" type="String">
            <hint-collection name="TEXTAREA" />
            <hint name="display-width">350</hint>
        </field>
        <field name="content" type="String">
            <hint-collection name="CLOB" />
            <sanitize content-type="text/html" modes="ALL" />
        </field>
        <field name="displayDate" type="Date" />
        <field name="allowPingbacks" type="boolean" />
        <field name="allowTrackbacks" type="boolean" />
        <field name="trackbacks" type="String">
            <hint-collection name="CLOB" />
        </field>
        <field name="coverImageCaption" type="String">
            <hint-collection name="TEXTAREA" />
        </field>
    </model>
</model-hints>
```

```
        <sanitize content-type="text/html" modes="ALL" />
    </field>
    <field name="coverImageFileEntryId" type="long" />
    <field name="coverImageURL" type="String">
        <hint-collection name="URL" />
    </field>
    <field name="smallImage" type="boolean" />
    <field name="smallImageFileEntryId" type="long" />
    <field name="smallImageId" type="long" />
    <field name="smallImageURL" type="String">
        <hint-collection name="URL" />
        <hint name="display-width">210</hint>
    </field>
    <field name="lastPublishDate" type="Date" />
    <field name="status" type="int" />
    <field name="statusByUserId" type="long" />
    <field name="statusByUserName" type="String" />
    <field name="statusDate" type="Date" />
</model>
<model name="com.liferay.blogs.model.BlogsStatsUser">
    <field name="statsUserId" type="long" />
    <field name="groupId" type="long" />
    <field name="companyId" type="long" />
    <field name="userId" type="long" />
    <field name="entryCount" type="int" />
    <field name="lastPostDate" type="Date" />
    <field name="ratingsTotalEntries" type="int" />
    <field name="ratingsTotalScore" type="double" />
    <field name="ratingsAverageScore" type="double" />
</model>
</model-hints>
```

Local Service

When configuring the service, you can define which kind of services are generated in the `service.xml`. There are two kinds of services available: the Local service and the Remote service.

The Local service is meant to be a service access-point within the same JVM and without any permission checks. The Local service is the layer where you call the persistence layer to retrieve and store data entities.

The Local service generation is enabled in `service.xml` :

```
<entity name="Assignment" local-service="true" ... >
```

Remote Service

The remote service serves two purposes. First, it's meant to be a façade layer for the local service, a place where you can implement permission checking. The other purpose is to provide JSON and SOAP APIs for the service.

If you are providing user-level access to the service, you should implement the remote service layer with permission checks and access the service through it. When you don't need permission checks, accessing the service through the local service should be preferred because of better performance.

Remote service generation for an entity is enabled in the entity definition in

`service.xml` :

```
<entity name="Assignment" remote-service="true" ... >
```

Service Implementation Classes

Service implementation classes are for overriding and customizing the generated service methods and for implementing your own business logic. They are the **only generated classes to be modified manually**. When you build and generate the service, the implementation classes are, depending on the entity configuration, created in the service module for:

- Local service
- Remote service
- Entity model class
- Entity finders

When implementing custom CRUD logic in the local or remote service implementation classes, the following naming and signature pattern is recommended:

- `{entity}ServiceImpl.add{entity}(userId, groupId, {all entity fields}, {serviceContext});`
 - returns created `{entity}`
- `{entity}ServiceImpl.update{entity}({primaryKey}, {all entity fields}, {serviceContext})`
 - returns updated `{entity}`
- `{entity}ServiceImpl.delete{entity}({primaryKey});`
 - returns deleted `{entity}`

As an example, in the Gradebook exercise, we'll use this pattern for example in the

`AssignmentLocalServiceImpl.updateAssignment() :`

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

    Assignment assignment = getAssignment(assignmentId);

    assignment.setModifiedDate(new Date());
    assignment.setTitleMap(titleMap);
    assignment.setDueDate(dueDate);
    assignment.setDescription(description);

    assignment = super.updateAssignment(assignment);

    return assignment;
}
```

Having the entity as a return value in the CRUD methods is mandatory when using the `@Indexable` annotations in integrating with the search framework.

Remember that the `buildService` task must be run whenever `service.xml` is modified or any of the method signatures in the model, service, or finder implementation classes have been changed. Thus, a typical workflow in creating the service layer is:

1. Define the model entities and finder queries in `service.xml`.
2. Run the `buildService` task.
3. Modify implementation classes.
4. Run the `buildService` task.
5. Modify the implementation classes.
6. Run the `buildService` task.
7. ...

"Silencing" Generated Service Methods

Sometimes it's useful to silence generated methods or method signatures to ensure the correct usage of the API. Below is an example of overriding a generated signature in a service implementation class:

```
@Override  
public class AssignmentLocalServiceImpl extends AssignmentLocalSe  
rviceBaseImpl {  
  
    public Assignment addAssignment(Assignment assignment) {  
        throw new UnsupportedOperationException(  
            "please use instead addAssignment(long groupId, long  
            userId, Map<Locale, String> titleMap," +  
            " String description, Date dueDate, ServiceContext)")  
    }  
}
```

Finders

Finders are methods for querying the database and are defined in `service.xml`. They are automatically cached and can be customized in the entity-specific finder implementation classes.

There are two kind of finders:

- **Finders:** work without permission-checking
- **Filtered finders:** provide permission-checking

The actual finder methods are created in the persistence classes. To access those via your service, you have to implement facade methods for them. Also note that entity **permissions have to be defined** in order to have filtered finders automatically created.

Finders in Liferay core services are using a naming pattern using the first letters of the finder parameters, separated by underscores. The following finder in the [Blogs application](#) is returning a collection of Blogs entries and accepting `groupId` and `status` as parameters:

```
<finder name="G_S" return-type="Collection">
    <finder-column name="groupId" />
    <finder-column name="status" />
</finder>
```

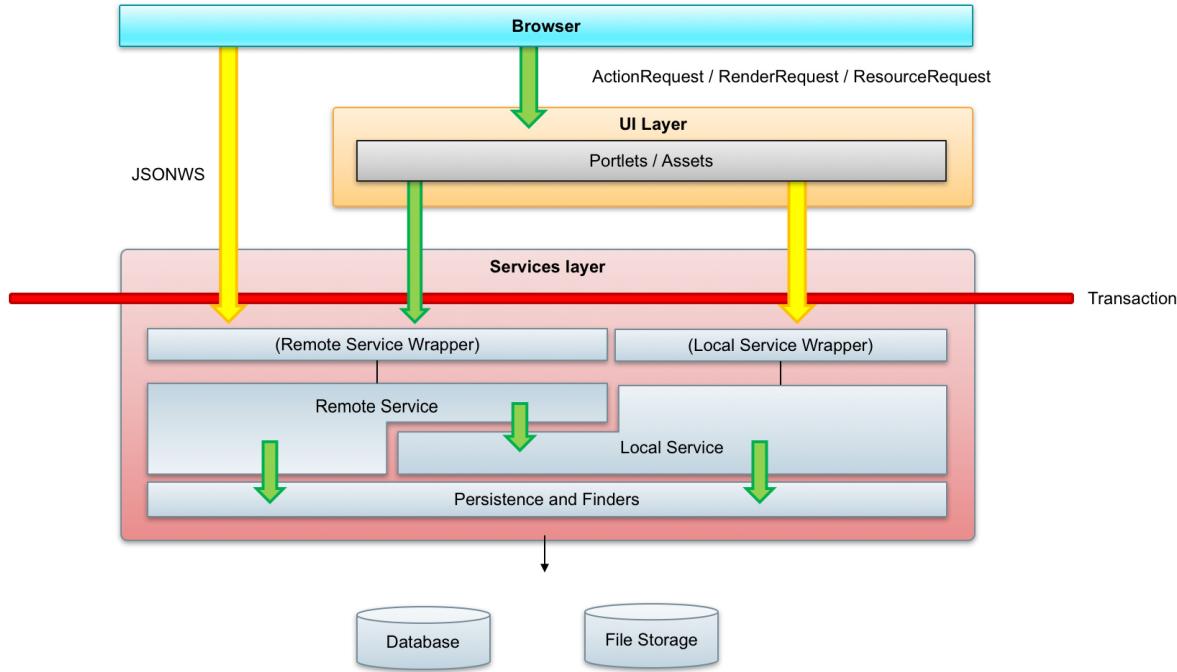
Finder methods are created in the persistence classes. To access those via your service, you have to implement façade methods for them. Here is an example from the [BlogsEntryPersistenceImpl](#):

```
@Override
public List<BlogsEntry> findByG_S(long groupId, int status) {
    return findByG_S(
        groupId, status, QueryUtil.ALL_POS, QueryUtil.ALL_POS, null);
}
```

Service Wrappers

When the `buildService` task is run, it not only generates implementation classes for all the model entities, it also generates so-called service wrapper APIs, which allow you to override your services in an external module. With this approach, you can also override Liferay core services.

The diagram below illustrates Service Builder design and architecture as well as the positioning of service wrappers:



Service Wrappers will be discussed in detail in *Module 10 - Customize the Service Layer*.

Service Context

When you work with Liferay services you'll often see a `ServiceContext` object in the method parameters. The `ServiceContext` is a wrapper object for contextual information, implementing the [Parameter Object](#) design pattern. It aggregates information necessary for features used throughout Liferay's portlets and services, such as:

- Actions
- Request parameters
- Classifications (tags and categories)
- Exceptions
- Scoping (company and group)
- Locale
- Request object
- Permission-related information

All the request parameters can be accessed with the `ServiceContext.getAttribute()` method.

See [Liferay Developer Network](#) for more information about the `ServiceContext`.

Service Builder Caching

Service Builder provides built-in caching support for all entities and finders defined in the service configuration. Cache implementation relies on [EHCache](#).

Caching is implemented on three levels:

- Hibernate
- Entity
- Finder

Hibernate cache has two layers: level 1 (L1) and level 2 (L2). **Level 1** is used to cache objects retrieved from the database within the current database session, which is usually tied to the invocation of Liferay's service layer within a single request.

Level 2 stores both database objects (Entity Cache) and results of queries (Query Cache) and is able to span across database sessions. Liferay provides custom Service Builder Entity and Query (Finder) caches, making the Hibernate L2 cache redundant. L2 cache is disabled by default, and unless you are accessing Hibernate code directly, there shouldn't be a reason to enable it.

Service Builder and OSGi

An important thing to notice is that, currently, the Service Builder-generated service classes are Spring beans and not OSGi service components. Although they are wired to the OSGi service registry, you can't use the OSGi `@Reference` annotation inside the Service Builder-generated classes. Making other Liferay services available in your custom service class should be done by referencing them in the `service.xml` configuration file or using the Spring `@ServiceReference` and `@BeanReference` annotations.

With 7.2, the Service Builder is relying on OSGi and Declarative Services but you can still choose to use the Spring approach. The dependency injector attribute is defined in the `service.xml` and can have either the `ds` value for Declarative Services or the `spring` value for Spring AOP. For new OSGi modules, the default value is `ds` and for the WARs, it's `spring`.

```
<service-builder dependency-injector="ds" package-path="com.liferay.training.gradebook">
```

For further reading on the Service Builder resources in Liferay Developer Network:

https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/service-builder

Knowledge Check

- _____ is a code generation tool that takes an _____ as an input and generates a _____ as an output.
- There are two kinds of services available:
 - _____
 - _____
- _____ are database-querying methods with caching support.
- Service _____ are the only classes meant to be _____.
- _____ allow you to override Service Builder-generated services. They can also be used to override core Liferay services.

Exercises

Create the Assignment Service

Exercise Goals

- Create a Liferay Service Builder Project using the *service-builder* template
- Define the Assignment entity
- Define service exceptions
- Final code review
- Build the service

Create a Liferay Service Builder Project

Option 1: Use the Command Line Blade tools

1. Open the *Command Line* shell in your Liferay Workspace `modules` folder.
2. Run command:

```
blade create -t service-builder -v 7.2 -p com.liferay.training.gradebook gradebook
```

3. Run Gradle refresh on the IDE.

Option 2: Use the Developer Studio Wizard

1. Launch the *Liferay Module Project* wizard in Developer Studio.
2. Use the following information for the first step:
 - **Project Name:** "gradebook"
 - **Build Type:** Gradle
 - **Liferay Version:** 7.2
 - **Project Template:** service-builder
3. Click *Next* and use the following information in the second step:

- **Package Name:** "com.liferay.training.gradebook"

4. Click *Finish* to close the wizard.

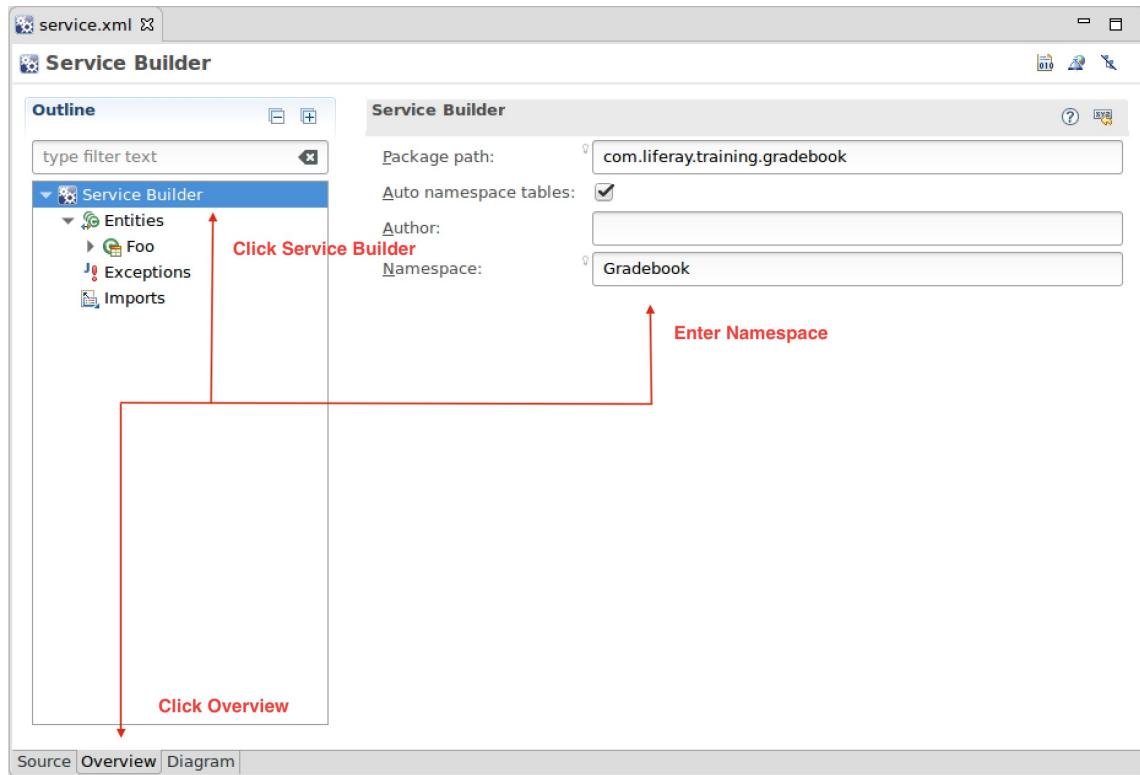
Define the Assignment Entity

`service.xml` is the main configuration file of a Service Builder project. It lets you define model entities, data sources, finder methods, and exceptions for your service.

You can customize `service.xml` with a graphical designer tool or edit the file's source code directly. We'll help you become familiar with both approaches.

First, define the **namespace** for our service. The namespace is used for prefixing the database tables Service Builder generates for the service:

1. Open the `service.xml` file in the *gradebook-service* folder.
2. Click on the *Overview* view.
3. Click *Service Builder* in the outline tree.
4. Enter "Gradebook" in the *Namespace*.



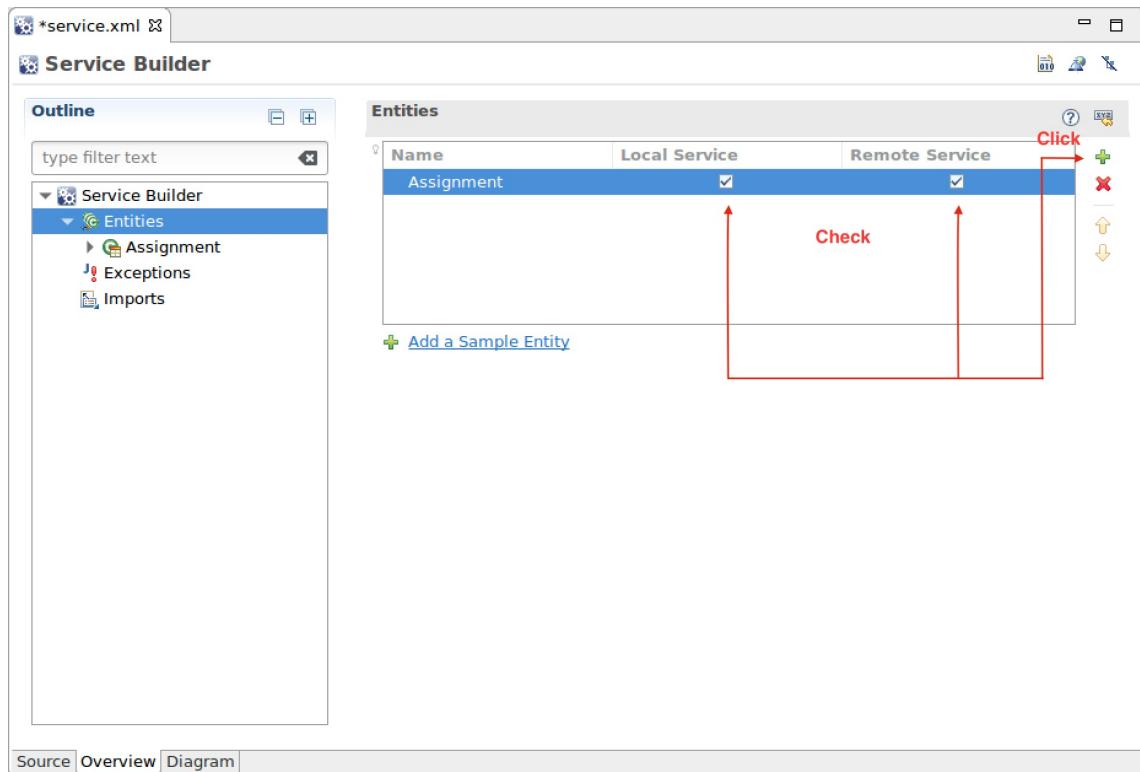
Delete the Example Entity

Delete the dummy entity created by the template. We don't need that for this exercise:

1. Right-click on the *Foo* entity in the outline tree and *delete* the entity.

Create the Assignment Entity

1. **Click** the green plus icon on the right side of the entities list to add a new entity.
2. **Enter** "Assignment" for the *Name* field.
3. **Check** both the *Local Service* and the *Remote Service*.



Define Assignment Columns

Define Assignment fields, which map to database model columns.

1. **Double-click** on the *Assignment* entity in the outline tree to open entity properties.
2. **Click** on the *Columns*.
3. **Click Add Default Columns** to add a default set of fields.

Take a look at the columns created. You'll see that a default primary key for the entity, `assignmentId`, was created automatically.

Let's add the *title* column:

1. **Click** the green plus sign on the right side of the columns list to add a new column.
2. **Enter** "title".
3. **Double-click** the *Type* column.

4. **Click** the browse icon on the right side of the field and select *String* type.

Let's now have a look at editing the `service.xml` in source mode instead of using the graphical designer.

Make the *title* field localizable:

Notice that you can use IDE's code-completion feature while editing the `service.xml` in source mode:

1. **Click** on the *Source* tab in the designer.
2. **Add** the `localized="true"` attribute for the *title* field.
 - The title field definition will now look like this:

```
<column name="title" type="String" localized="true" />
```

3. **Add** the rest of the Assignment's columns after the *title* column:

```
<column name="description" type="String" localized="true" />
<column name="dueDate" type="Date" />
```

Define the Default Sort Order

Define the default sort order for Assignment entities:

1. **Add** the following snippet after the *column* definitions:

```
<!-- Order -->

<order by="asc">
    <order-column name="title" />
</order>
```

Define the Finder Methods

Finders are methods for querying entities by columns. For now, we'll just need to be able to find items by *groupId*:

1. **Add** the following snippet after the *order* definition:

```
<!-- Finders -->

<!-- Find by groupId -->

<finder name="GroupId" return-type="Collection">
    <finder-column name="groupId"></finder-column>
</finder>
```

Define the Entity References

References define entity services injected in our service classes. This helps to keep the database calls inside a single transaction.

We need the Group services and Liferay Asset services for integrating to the Liferay Asset framework for later exercise steps.

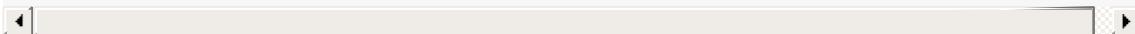
1. **Add** the following reference definitions after the *finder* definitions:

```
<!-- Reference to Group entity service -->

<reference entity="Group" package-path="com.liferay.portal"></reference>

<!-- Entity services needed for the integration to Asset framework -->

<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>
```



Define the Service Exceptions

Define an exception for validation purposes:

1. **Add** the following code snippet after the closing tag of *entity*:

```
<!-- Exceptions -->

<exceptions>
    <exception>AssignmentValidation</exception>
</exceptions>
```

Final Code Review

Use IDE's automatic code formatting to fix indents and spacing. The final `service.xml` will look like this:

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder
7.2.0//EN" "http://www.liferay.com/dtd/liferay-service-builder_7
_2_0.dtd">

<service-builder dependency-injector="ds" package-path="com.liferay.training.gradebook">
    <namespace>Gradebook</namespace>

        <entity local-service="true" name="Assignment" remote-service=
"true" uuid="true">

            <!-- PK fields -->

                <column name="assignmentId" primary="true" type="long"></
column>

            <!-- 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="title" type="String" localized="true"></col
umn>

<column name="description" type="String" localized="true"
/>
<column name="dueDate" type="Date" />

<!-- 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.porta
l"></reference>

<!-- Entity services needed for the integration to Asset
framework -->

<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>

```



Build the Service

There's only one step left: to build the service and generate service classes. When you run the `buildService` Gradle task, the following items are generated:

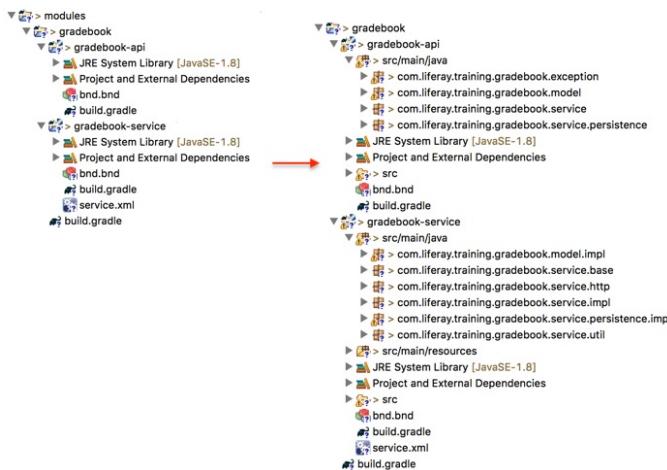
1. Database schema for the service (committed at module deploy time)
2. Persistence and caching
3. Local and remote service APIs

Remember that you have to rebuild services whenever you edit the `service.xml`.

Run the `buildService` Gradle task:

1. **Expand** the `training-workspace/modules/gradebook/build` in the *Gradle Tasks* panel.
2. **Execute** the `buildService` task.

The Assignment services have now been generated, and your project tree will look like this:



Notice that you can run the service generation task from several places:

- Using the project's `buildService` task on the *Gradle Tasks* panel

- The *Overview* panel of the `service.xml` designer
- The Context menu of the gradebook-service project
- From the *Command Line* using the Liferay Workspace `gradlew` script

Add Missing Dependencies

The autogenerated classes depend on packages, which aren't included in the service module's default dependency configuration.

1. **Configure** the dependencies in the service module's `build.gradle` file as follows:

```
dependencies {  
    compileOnly group: "com.liferay", name: "com.liferay.petra.lang"  
    compileOnly group: "com.liferay", name: "com.liferay.petra.string"  
    compileOnly group: "com.liferay", name: "com.liferay.portal.apop.api"  
    compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"  
    compileOnly group: "org.osgi", name: "org.osgi.annotation.versioning"  
    compileOnly group: "org.osgi", name: "org.osgi.core"  
    compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"  
    compileOnly project(":modules:gradebook:gradebook-api")  
}  
  
buildService {  
    apiDir = "../gradebook-api/src/main/java"  
}  
  
group = "com.liferay.training.gradebook"
```

Exercises

Implement Assignment Local Service

Exercise Goals

- Implement `addAssignment()`
- Implement `updateAssignment()`
- Implement the finder methods
- Silence generated methods
- Do a final code review
- Rebuild the service

Implement `addAssignment()`

Before implementing the method for adding assignments, open the local service base class `AssignmentLocalServiceBaseImpl` and take a look at the generated `addAssignment()` method. This method doesn't automatically generate an ID, set the audit fields (like creation or modification date), or validate the entity:

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

    return assignmentPersistence.update(assignment);
}
```

Create an overload for `addAssignment()` to take care of these tasks:

1. Open the `AssignmentLocalServiceImpl`. The empty class looks like this:

```
/*
 * 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.training.gradebook.service.base.AssignmentLocalServiceBaseImpl;

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

/*
 * The implementation of the assignment local service.
 *
 * <p>
 * All custom service methods should be put in this class. Whenever methods are added, rerun ServiceBuilder to copy their definitions into the <code>com.liferay.training.gradebook.service.AssignmentLocalService</code> interface.
 *
 * <p>
```

```

    * This is a local service. Methods of this service will not
    have security checks based on the propagated JAAS credentials
    because this service can only be accessed from within the same
    VM.
    *
    * </p>
    *
    * @author Brian Wing Shun Chan
    * @see AssignmentLocalServiceBaseImpl
    */
    @Component(
        property = "model.class.name=com.liferay.training.gradebook.model.Assignment",
        service = AopService.class
    )
    public class AssignmentLocalServiceImpl extends AssignmentLocalServiceBaseImpl {

        /*
         * NOTE FOR DEVELOPERS:
         *
         * Never reference this class directly. Use <code>com.liferay.training.gradebook.service.AssignmentLocalService</code>
         * via injection or a <code>org.osgi.util.tracker.ServiceTracker</code> or use <code>com.liferay.training.gradebook.service.AssignmentLocalServiceUtil</code>.
         */
    }

```

2. **Implement** the `addAssignment()` in the class as follows:

```

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

    // 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 assignment. This doesn't yet persist the entity.

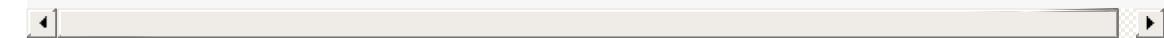
Assignment assignment = createAssignment(assignmentId);

// Populate fields.

assignment.setCompanyId(group.getCompanyId());
assignment.setCreateDate(serviceContext.getCreateDate(new Date()));
assignment.setDueDate(dueDate);
assignment.setDescriptionMap(descriptionMap);
assignment.setGroupId(groupId);
assignment.setModifiedDate(serviceContext.getModifiedDate(new Date()));
assignment.setTitleMap(titleMap);
assignment.setUserId(userId);
assignment.setUserName(user.getScreenName());

// Persist assignment to database.

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



Implement updateAssignment

1. **Create** an overload for the `updateAssignment()` :

```

public Assignment updateAssignment(
    long assignmentId, Map<Locale, String> titleMap, Map<Loca
le, String> descriptionMap,
    Date dueDate, ServiceContext serviceContext)
throws PortalException {

    // Get the Assignment by id.

    Assignment assignment = getAssignment(assignmentId);

    // Set updated fields and modification date.

    assignment.setModifiedDate(new Date());
    assignment.setTitleMap(titleMap);
    assignment.setDueDate(dueDate);
    assignment.setDescriptionMap(descriptionMap);

    assignment = super.updateAssignment(assignment);

    return assignment;
}

```

Implement the Finder Methods

Defining finders in `service.xml` automatically creates the corresponding methods in the persistence classes, but we cannot access those directly from the controller layer and have to implement façades in the service implementation class.

1. **Implement** the finder methods as follows:

```

public List<Assignment> getAssignmentsByGroupId(long groupId)
{

    return assignmentPersistence.findByGroupId(groupId);
}

public List<Assignment> getAssignmentsByGroupId(
    long groupId, int start, int end) {

```

```

        return assignmentPersistence.findByGroupId(groupId, start
, end);
    }

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

    return assignmentPersistence.findByGroupId(
        groupId, start, end, orderByComparator);
}

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

    return assignmentLocalService.dynamicQuery(
        getKeywordSearchDynamicQuery(groupId, keywords), star
t, end,
        orderByComparator);
}

public long getAssignmentsCountByKeywords(long groupId, Strin
g keywords) {
    return assignmentLocalService.dynamicQueryCount(
        getKeywordSearchDynamicQuery(groupId, keywords));
}

private DynamicQuery getKeywordSearchDynamicQuery(
    long groupId, String keywords) {

    DynamicQuery dynamicQuery = dynamicQuery().add(
        RestrictionsFactoryUtil.eq("groupId", groupId));

    if (Validator.isNotNull(keywords)) {
        Disjunction disjunctionQuery =
            RestrictionsFactoryUtil.disjunction();

        disjunctionQuery.add(
            RestrictionsFactoryUtil.like("title", "%" + keywo

```

```

        rds + "%"));
        disjunctionQuery.add(
            RestrictionsFactoryUtil.like(
                "description", "%" + keywords + "%"));
        dynamicQuery.add(disjunctionQuery);
    }

    return dynamicQuery;
}

```

For the sake of this exercise, we introduced a custom `getAssignmentsByKeywords()` method here, which we will use on the user interface later for searching. This method is using [Dynamic Queries](#), which allow you to query the database with custom SQL. Note that this specific query wouldn't work well with localized fields, which are stored in xml.

"Silence" the Generated Method

Sometimes it's practical to silence generated methods to ensure correct API usage. Override and "silence" the generated `addAssignment()` and `updateAssignment()` method signatures, which we replaced with our overrides before:

```

@Override
public Assignment addAssignment(Assignment assignment) {
    throw new UnsupportedOperationException("Not supported.");
}

@Override
public Assignment updateAssignment(Assignment assignment) {
    throw new UnsupportedOperationException(
        "Not supported.");
}

```

Do a Final Code Review

1. **Resolve** missing imports.
2. **Fix** indents and spacing by using automatic code formatting.

The final `AssignmentLocalServiceImpl.java` class will look like this:

```
/*
 * 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 Li
 * cense for more
 * details.
 */

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

import com.liferay.portal.aop.AopService;
import com.liferay.portal.kernel.dao.orm.Disjunction;
import com.liferay.portal.kernel.dao.orm.DynamicQuery;
import com.liferay.portal.kernel.dao.orm.RestrictionsFactoryUtil;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.model.Group;
import com.liferay.portal.kernel.model.User;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.util.OrderByComparator;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.base.AssignmentLoca
lServiceBaseImpl;
```

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

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

/**
 * The implementation of the assignment local service.
 *
 * <p>
 * All custom service methods should be put in this class. Whenever methods are
 * added, rerun ServiceBuilder to copy their definitions into the
 * <code>com.liferay.training.gradebook.service.AssignmentLocalSe
rvice</code>
 * interface.
 *
 * <p>
 * This is a local service. Methods of this service will not have
 * security
 * checks based on the propagated JAAS credentials because this s
ervice can only
 * be accessed from within the same VM.
 * </p>
 *
 * @author Brian Wing Shun Chan
 * @see AssignmentLocalServiceBaseImpl
 */

@Component(
    property = "model.class.name=com.liferay.training.gradebook.m
odel.Assignment",
    service = AopService.class
)
public class AssignmentLocalServiceImpl extends AssignmentLocalSe
rviceBaseImpl {

    /*
     * NOTE FOR DEVELOPERS:
    */
}
```

```
*  
 * Never reference this class directly. Use  
 * <code>com.liferay.training.gradebook.service.AssignmentLoc  
 alService</code>  
 * via injection or a <code>org.osgi.util.tracker.ServiceTrac  
 ker</code> or use  
 * <code>com.liferay.training.gradebook.service.AssignmentLoc  
 alServiceUtil</code>  
 * >  
 */  
  
public Assignment addAssignment(long groupId, Map<Locale, Str  
ing> titleMap, Map<Locale, String> descriptionMap,  
        Date dueDate, ServiceContext serviceContext) throws P  
ortalException {  
  
    // 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(Assignm  
ent.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.setDescriptionMap(descriptionMap);
```

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

        // Persist assignment to database.

        return super.addAssignment(assignment);
    }

    public Assignment updateAssignment(long assignmentId, Map<Loc
ale, String> titleMap,
        Map<Locale, String> descriptionMap, Date dueDate, Ser
viceContext serviceContext) throws PortalException {

        // Get the Assignment by id.

        Assignment assignment = getAssignment(assignmentId);

        // Set updated fields and modification date.

        assignment.setModifiedDate(new Date());
        assignment.setTitleMap(titleMap);
        assignment.setDueDate(dueDate);
        assignment.setDescriptionMap(descriptionMap);

        assignment = super.updateAssignment(assignment);

        return assignment;
}

public List<Assignment> getAssignmentsByGroupId(long groupId)
{
    return assignmentPersistence.findByGroupId(groupId);
}

public List<Assignment> getAssignmentsByGroupId(long groupId,
```

```
int start, int end) {

    return assignmentPersistence.findByGroupId(groupId, start,
, end);
}

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

    return assignmentPersistence.findByGroupId(groupId, start
, end, orderByComparator);
}

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

    return assignmentLocalService.dynamicQuery(
        getKeywordSearchDynamicQuery(groupId, keywords), star
t, end,
        orderByComparator);
}

public long getAssignmentsCountByKeywords(long groupId, Strin
g keywords) {
    return assignmentLocalService.dynamicQueryCount(
        getKeywordSearchDynamicQuery(groupId, keywords));
}

private DynamicQuery getKeywordSearchDynamicQuery(
long groupId, String keywords) {

    DynamicQuery dynamicQuery = dynamicQuery().add(
        RestrictionsFactoryUtil.eq("groupId", groupId));

    if (Validator.isNotNull(keywords)) {
        Disjunction disjunctionQuery =
            RestrictionsFactoryUtil.disjunction();

```

```
        disjunctionQuery.add(
            RestrictionsFactoryUtil.like("title", "%" + keywo
rds + "%"));
        disjunctionQuery.add(
            RestrictionsFactoryUtil.like(
                "description", "%" + keywords + "%"));
        dynamicQuery.add(disjunctionQuery);
    }

    return dynamicQuery;
}

@Override
public Assignment addAssignment(Assignment assignment) {

    throw new UnsupportedOperationException("Not supported.");
;

}

@Override
public Assignment updateAssignment(Assignment assignment) {

    throw new UnsupportedOperationException("Not supported.");
;

}

}
```

Rebuild the Service

1. Run the `buildService` Gradle task to regenerate the service.

Exercises

Implement Assignment Remote Service

Exercise Goals

- Declare dependencies
- Implement the façade methods for the local service in the `AssignmentServiceImpl.java`
- Do a final code review
- Rebuild and deploy the service

Declare Dependencies

For our implementation, we need to make the Servlet and Portlet API available:

1. Open the `build.gradle` of `gradebook-service`.
2. Implement the new dependencies as follows:

```
compileOnly group: "javax.portlet", name: "portlet-api"  
compileOnly group: "javax.servlet", name: "javax.servlet-api"
```

Implement the Façade Methods

1. Open the `AssignmentServiceImpl.java` class. The empty class looks like this:

```
/**  
 * 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.  
 *  
 * You should have received a copy of the GNU Lesser General Public  
 * License along with this library; if not, write to the Free Software  
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301  
 * USA  
 */
```

```
ished 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 us
eful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANT
ABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Publi
c License for more
 * details.
 */

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

import com.liferay.portal.aop.AopService;
import com.liferay.training.gradebook.service.base.Assignment
ServiceBaseImpl;

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

/**
 * The implementation of the assignment remote service
 *
 * <p>
 * All custom service methods should be put in this class. Wh
enever methods are added, rerun ServiceBuilder to copy their d
efinitions into the <code>com.liferay.training.gradebook.servi
ce.AssignmentService</code> interface.
 *
 * <p>
 * This is a remote service. Methods of this service are expe
cted to have security checks based on the propagated JAAS cred
entials because this service can be accessed remotely.
 * </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 AssignmentServiceB
aseImpl {

    /*
     * NOTE FOR DEVELOPERS:
     *
     * Never reference this class directly. Always use <code>
     * com.liferay.training.gradebook.service.AssignmentServiceUtil</
     * code> to access the assignment remote service.
     */
}
}
```

2. **Implement** the façade methods in the class as follows:

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

    return assignmentLocalService.addAssignment(
        groupId, titleMap, descriptionMap, dueDate, serviceCo
ntext);
}

public Assignment deleteAssignment(long assignmentId)
throws PortalException {

    Assignment assignment =
        assignmentLocalService.getAssignment(assignmentId);

    return assignmentLocalService.deleteAssignment(assignment
}
```

```
);

}

public Assignment getAssignment(long assignmentId)
    throws PortalException {

    Assignment assignment =
        assignmentLocalService.getAssignment(assignmentId);

    return assignment;
}

public List<Assignment> getAssignmentsByGroupId(long groupId)
{
    return assignmentPersistence.findByGroupId(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, Map<Locale, String> descriptionMap,
    Date dueDate, ServiceContext serviceContext)
    throws PortalException {
```

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

3. **Resolve** missing imports.

Final Code Review

The complete files will look like this:

```
dependencies {
    compileOnly group: "com.liferay", name: "com.liferay.petra.lang"
    compileOnly group: "com.liferay", name: "com.liferay.petra.string"
    compileOnly group: "com.liferay", name: "com.liferay.portal.apop.api"
    compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"
    compileOnly group: "javax.portlet", name: "portlet-api"
    compileOnly group: "javax.servlet", name: "javax.servlet-api"
    compileOnly group: "org.osgi", name: "org.osgi.annotation.versioning"
    compileOnly group: "org.osgi", name: "org.osgi.core"
    compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"
    compileOnly project(":modules:gradebook:gradebook-api")
}

buildService {
    apiDir = "../gradebook-api/src/main/java"
}

group = "com.liferay.training.gradebook"
```

```
/**
 * 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 Li  
 cense 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.service.ServiceContext;  
import com.liferay.portal.kernel.util.OrderByComparator;  
import com.liferay.training.gradebook.model.Assignment;  
import com.liferay.training.gradebook.service.base.AssignmentServ  
iceBaseImpl;  
  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
import java.util.Map;  
  
import org.osgi.service.component.annotations.Component;  
  
/**  
 * The implementation of the assignment remote service.  
 *  
 * <p>  
 * All custom service methods should be put in this class. Whenev
```

```
er methods are added, rerun ServiceBuilder to copy their definitions into the <code>com.liferay.training.gradebook.service.AssignmentService</code> interface.

/*
 * <p>
 * This is a remote service. Methods of this service are expected to have security checks based on the propagated JAAS credentials because this service can be accessed remotely.
 * </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.gradebook.service.AssignmentServiceUtil</code> to access the assignment remote service.
     */
    public Assignment addAssignment(
        long groupId, Map<Locale, String> titleMap, Map<Locale, String> descriptionMap,
        Date dueDate, ServiceContext serviceContext)
        throws PortalException {

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

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

    Assignment assignment =
        assignmentLocalService.getAssignment(assignmentId);

    return assignmentLocalService.deleteAssignment(assignment
);
}

public Assignment getAssignment(long assignmentId)
    throws PortalException {

    Assignment assignment =
        assignmentLocalService.getAssignment(assignmentId);

    return assignment;
}

public List<Assignment> getAssignmentsByGroupId(long groupId)
{

    return assignmentPersistence.findByGroupId(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.getAssignmentsCountByKeyw
ds(
    groupId, keywords);
```

```
}

public Assignment updateAssignment(
    long assignmentId, Map<Locale, String> titleMap, Map<Loca
le, String> descriptionMap,
    Date dueDate, ServiceContext serviceContext)
    throws PortalException {

    return assignmentLocalService.updateAssignment(
        assignmentId, titleMap, descriptionMap, dueDate, serv
iceContext);
}
```

}

Rebuild and Deploy the Service

Now it's time to deploy our service to the server.

1. **Run** the `buildService` Gradle task to regenerate the service.
2. **Start** the server if it's not running.
3. **Drag** the `gradebook-api` and `gradebook-service` modules onto the Liferay server to deploy the modules.

You should see a success log message if modules were deployed successfully:

```
2019-03-20 11:31:59.667 INFO [pipe-start 984][BundleStartStopLog
ger:39] STARTED com.liferay.training.gradebook.api_1.0.0 [984]
2019-03-20 11:32:02.573 INFO [pipe-start 985][BundleStartStopLog
ger:39] STARTED com.liferay.training.gradebook.service_1.0.0 [985
]
```

Create the Presentation Layer

In modular Liferay applications, the presentation layer is usually located in what's called the *web module*. For example, in the native Blogs application, this module is called [blogs-web](#). In terms of the MVC pattern, the web module typically contains both the view (user interface) and the controller (portlet) layers.

You can use a variety of technologies to develop Liferay portlets. Module templates are provided for example for:

- **mvc-portlet:** Liferay MVC portlet
- **freemarker-portlet:** portlet using Freemarker template language
- **npm-angular-portlet:** Angular portlet
- **npm-react-portlet:** React portlet
- **npm-vuejs-portlet:** VueJS portlet
- **spring-mvc-portlet:** Spring WAR style portlet
- **war-mvc-portlet:** MVC WAR style portlet

You can also use JSF, Vaadin or with portlet 3.0, supported since Liferay DXP 7.1, or take advantage of Java EE CDI with Bean Portlets. There are plenty of examples available in [Liferay Blade Samples](#) and [Liferay Developer Network resources](#).

The implementation steps and their order of execution depend on the chosen technology and approach. For the Gradebook exercise, we will, in general, be using the following approach:

1. Create the web module.
2. Implement portlet actions using MVC commands.
3. Implement portlet JSPs using tag libraries.
4. Implement portlet validation and feedback.
5. Add localization resources.
6. Add CSS resources.
7. Add JavaScript resources.

Let's discuss the steps and concepts related to the user interface of our Gradebook application.

Creating the Web Module

Liferay provides several module templates for building the presentation layer, like:

- **mvc portlet:** a module template with a sample Liferay MVC portlet component, localization resources, and JSP files
- **freemarker-portlet:** a module template for the user interface with FreeMarker templating language and FreeMarker portlet back-end. The FreeMarkerPortlet class is an extension of Liferay MVC portlet.
- **NPM portlets:** several Liferay MVC portlet-based templates for building the user interface with JavaScript frameworks like Angular and React; package management with NPM
- **spring-mvc-portlet:** for building a Spring MVC portlet
- **war-mvc-portlet:** a template for legacy WAR-style portlets

In addition to these, JSF and [Vaadin](#) portlet-type user interfaces are supported out of the box.

Implementing Portlet Actions with MVC Commands

The interaction between a portlet back-end and a user is handled by portlet lifecycle methods as described in *Module 5 - Java Standard Portlet*.

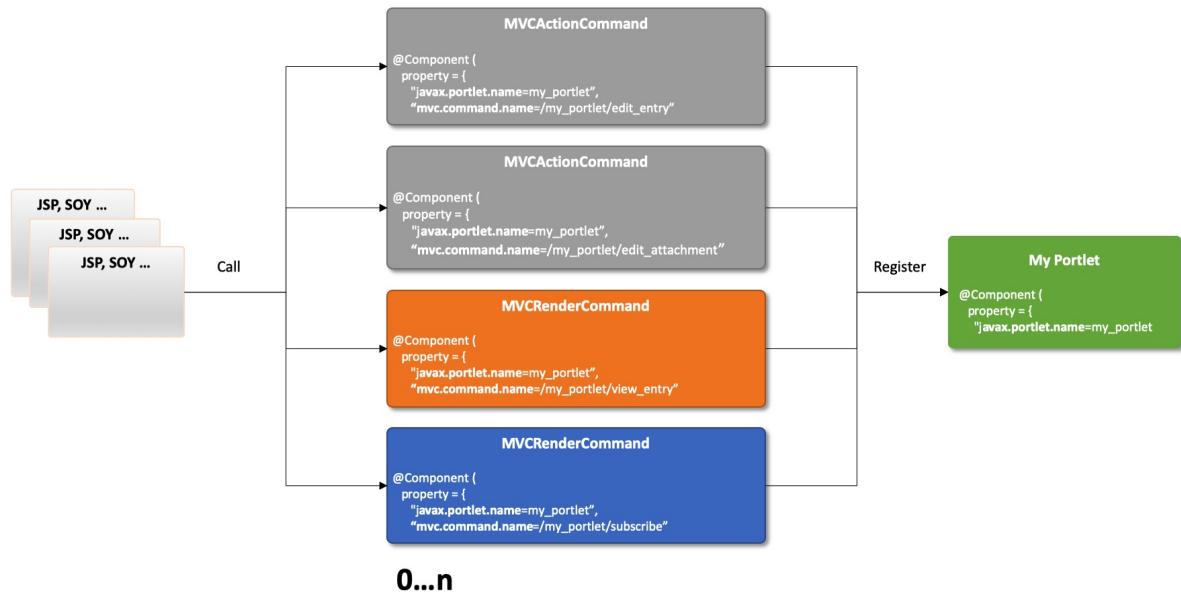
MVC commands are Liferay-provided, portlet lifecycle handler components meant to be used in conjunction with Liferay MVC portlets. They reduce the need for boilerplate portlet coding and provide a more modular application design compared to legacy-style lifecycle handlers.

MVC commands are single class OSGi components, responsible for handling an action defined by components in the `mvc.command.name` property. Generally, three qualities wire an MVC command to the right action in the right place:

1. Wiring to a certain portlet lifecycle (render, action, resource)
2. Responding to a certain command, defined in component properties with `mvc.command.name`
3. Registration to a certain portlet via the `javax.portlet.name` component property

Notice that a single MVC command component can respond to multiple commands and also register to multiple portlets.

The following diagram illustrates the architecture:



By convention, the MVC command classes are named by the following pattern, corresponding to the portlet lifecycle:

- *MVCRenderCommand.java
- *MVCActionCommand.java
- *MVCResourceCommand.java

With this pattern, an MVC command responsible for rendering assignments list view could become `ViewAssignmentsMVCRenderCommand`.

MVC Render Commands

MVC Render Commands handle portlet's *render* phase. They are called by setting the `mvcrendercommandname` parameter in the calling URL:

```

<portlet:renderURL var="viewEntryUrl">
    <portlet:param name="mvcrendercommandname" value="/my_portlet
/view_entry" />
    <portlet:param name="entryId" value="<%= String.valueOf(entry
.getEntryId()) %>" />
</portlet:renderURL>

<a href="<%= viewEntryUrl %>">Click here to view the entry</a>

```

MVC Action Commands

MVC action commands handle the portlet's *action* phase. Actions are typically form submits that trigger events, like entity update, on the model layer. The action to respond is defined in the calling URL's `name` parameter.

```
<portlet:actionURL name="/my_portlet/edit_entry" var="editEntryURL" />

<aui:form action="<% editEntryURL %>" method="post" name="fm">
```

MVC Resource Commands

MVC Resource Commands handle the *resource serving* phase. As the resource serving lifecycle phase doesn't invoke the render phase, it's typically used for operations that don't need full page refresh. Such use cases can be, for example:

- Updating a list with Ajax call
- Subscription actions
- Captcha checking

```
<portlet:resourceURL id="/login/captcha" var="captchaURL" />
```

Using Tag Libraries

Tag libraries are collections of user interface components called *tags* for JSP development. From a user interface design perspective, they allow a clean separation between the look-and-feel and business logic. Tag libraries can significantly reduce development time and remove boilerplate coding.

Tags inside a tag library are JSP markup elements, which are replaced by a respective markup at render times. Tags available in a library can be browsed in the library's descriptor TLD file (inside taglib JAR).

Liferay-provided libraries contain tags for the most common user interface components (like forms) and are designed to be responsive.

Below is an example of using a `liferay-ui` tag library for showing user details and rendered HTML. Notice the required library TLD declaration, which typically resides in an `init.jsp` file. Also notice the namespacing of the tag library, using the `prefix` attribute:

init.jsp

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

```html view.jsp

```
Rendered HTML
```
<div class="profile-header">
    <div class="nameplate">
        <div class="nameplate-field">
            <div class="user-icon-color-0 user-icon user-icon-default">
                <span>TT</span>
            </div>
        </div>
        <div class="nameplate-content">
            <div class="heading4">
                <a href="http://localhost:8080/web/test"> Test Test </a>
            </div> </div> <div class="nameplate-content">
        </div>
    </div>
</div>
```

Standard Tag Libraries

The *Portlet Standard* libraries are:

- **portlet**: standard portlet JSR 168 tag library (overrun in Liferay's `portlet_2_0` library)
- **portlet_2_0**: standard portlet JSR 286 tag library (included in Liferay's `liferay_portlet_2_0_ext`)

- **portlet_3_0:** standard portlet JSR 362 tag library
- **JSTL tag libraries**

Standard JSTL Libraries contain general purpose tags, for example, for:

- Variables, conditionals (if-else), loops
- Formatting strings, dates, and numbers
- Basic string functions (replace, join, lowercase, etc.)
- Parsing XML
- Showing date

Below is an example of standard JSTL tags usage (in `c` namespace):

```
<c:choose>
    <c:when test="#{themeDisplay.isSignedIn()}">
        <p>You appear to be signed in. Your groups: </p>
        <ul>
            <c:forEach items="#{themeDisplay.getUser().getGroups()}"
var="group">
                <li><c:out value = "${group.getName(locale)}"/></li>
            </c:forEach>
        </ul>
    </c:when>
    <c:otherwise>
        <p>You are not signed in.
    </c:otherwise>
</c:choose>
```

The standard JSR-362 (portlet_3_0) library provides portlet lifecycle url generation and portlet namespacing. Below are examples of creating an action url and render url with the portlet 3.0 tag library:

TLD_declaration

```
<%@ taglib uri="http://xmlns.jcp.org/portlet_3_0" prefix="portlet"
%>
```



actionURL

```
<portlet:actionURL name="/gradebook/assignment/delete" var="deleteAssignmentURL">
    <portlet:param name="redirect" value="${currentURL}" />
    <portlet:param name="assignmentId" value="${assignment.assignmentId}" />
</portlet:actionURL>
```

renderURL

```
<portlet:renderURL var="viewSubmissionsURL">
    <portlet:param name="mvcRenderCommandName" value="/gradebook/submissions/view" />
    <portlet:param name="redirect" value="${currentURL}" />
    <portlet:param name="assignmentId" value="${assignment.getAssignmentId()}" />
</portlet:renderURL>
```

Liferay Tag Libraries

Liferay provides a rich set of its own libraries, like:

- **liferay-theme**: provides theme components
- **liferay-portlet**: a wrapper for standard portlet_2_0 library
- **liferay-ui**: contains numerous display tags like alert, icons, search-container
- **liferay-util**: provides page-level tags like includes
- **aui**: Responsive Alloy UI tags for forms and containers
- **liferay-security**: provides doAsURL and permissionURL
- **clay**: set of tags for creating Liferay Clay UI components
- **chart**: charting and data modeling tags

The Clay library supersedes the AUI library of many parts, leveraging the Liferay Clay framework and providing tags, for example, for:

- Alerts
- Badges
- Buttons
- Cards
- Dropdown Menus and Action Menus

- Form Elements
- Icons
- Labels and links
- Navigation Bars
- Progress Bars
- Stickers

Below is an example of using the Clay library:

init.jsp

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

view.jsp

```
<clay:stripe  
    message="This is a success message."  
    style="success"  
    title="Success"  
/>
```

The code above renders as:

 Success This is a success message.



See [Developer Network](#) for more information about Liferay tag libraries.

Implementing Validation and Feedback

In real world application design, it's important to have control over both user input and output. A robust validation is done on multiple layers, and it never relies solely on the user interface.

When implementing validation and feedback, it's good to be aware of Liferay's utility classes. Liferay provides utility classes for validation but other common tasks, like string manipulation, date formatting and parameter handling. Many of the utilities can be found in the [com.liferay.portal.kernel.util](#) package and in the [Petra libraries](#).

Input Validation

There are many good reasons to implement multi-level input validation and user interface feedback in your applications:

1. **Security:** to protect against malicious input, no user input should be allowed to enter the model layer without validation
2. **Usability:** early validation and feedback provides a better user experience
3. **Resource usage:** validation on a user interface reduces faulty form submissions and server load

Client side input validation is provided with the `aui-validator` tag and Twitter Bootstrap.

The following tags support validation with `<aui-validator>` tag:

- `<aui:input>`
- `<aui:select>`
- `<liferay-ui:input-date>`
- `<liferay-ui:input-search>`

Below is a simple example of using the `<aui-validator>` tag with `<aui-input>` tag:

```
<aui:form>
    <aui:input name="title" >
        <aui:validator name="required" errorMessage="Please enter
the title."/>
    </aui:input>
<aui:form>
```

A custom validator function can be provided for the `<aui-validator>` tag. The validator function has to return true or false:

```
<aui:input name="title">
    <aui:validator errorMessage="you-must-specify-a-file-or-a-tit
le" name="custom">
        function(val, fieldNode, ruleValue) {
            return ((val != '') || A.one('#<portlet:namespace />f
ile').val() != '');
        }
    </aui:validator>
```

```
</aui:input>
```

Liferay's user interface relies on Twitter Bootstrap to support its validation methods. Below is an example of using the `required` attribute:

```
<form data-toggle="validator" role="form">
    <div class="form-group">
        <label for="inputName" class="control-label">Name</label>
        <input type="text" class="form-control" id="inputName" placeholder="Your Name" required>
    </div>
    <div class="form-group">
        <button type="submit" class="btn btn-primary">Submit</button>
    </div>
</form>
```

On the **back-end** side, the [Validator](#) utility class provides methods for common validation tasks:

```
String name = ParamUtil.getString(request, "title");
If (Validator.isNull(name)) {
    SessionErrors.add(request, "title-empty");
    return false;
}
```

Note on AntiSamy

Although not a validation functionality as such, there is an additional security module that protects against malicious user input. The **AntiSamy** module leverages the OWASP AntiSamy library, processing user input on form submit and stripping away all the HTML elements and content not explicitly allowed. The module is configurable through *Control Panel → System Settings → Foundation → AntiSamy Sanitizer*.

Output Validation

In addition to input validation, an output validation for malicious content should be done to prevent malicious code from executing. Usually, this means escaping content prior to displaying.

On the client side, escaping can be done with the standard JSTL library:

```
<c:out escapeXML="true">${bodyText}</c:out>
```

On the server side, the Liferay HTMLUtil class can be used:

```
request.setAttribute("bodyText", HtmlUtil.escape(bodyText));
```

Showing Feedback

In Liferay portlet JSP applications, the feedback from the back-end to the user interface is often transported with [SessionErrors](#) and [SessionMessages](#) objects. Below is an example demonstrating 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:

Portlet class

```
String name = ParamUtil.getString(request, "title");
If (Validator.isNull(name)) {
    SessionErrors.add(request, "titleEmpty");
    return false;
}

if (SessionErrors.isEmpty()) {
    SessionMessages.add(request, "assignmentUpdatedSuccessfully")
;
}
```

Language.properties

```
assignment-updated-successfully=Assignment was updated successfully
```

```
please-enter-title=Please enter the title.
```

view.jsp

```
<liferay-ui:success key="assignmentUpdatedSuccessfully" message="assignment-updated-successfully" />
<liferay-ui:error key="titleEmpty" message="please-enter-title" />
```



This renders on error as:

```
Plese enter the title.
```

Hiding Default Messages

Liferay sets default success messages for successful portlet actions. These messages can be hidden with portlet component properties:

```
@Component(
    immediate = true,
    property = {
        ...
        "javax.portlet.init-param.add-process-action-success-acti
on=false",
        ...
    },
    service = Portlet.class
)
```

Common Guidelines for Implementing Validation

- Establish validation on both the client and server side.
 - User interface validation is not for securing but for usability.
- Establish control over all input.
- Establish control over all output; escape the output on the user interface.
- Escaping of user-provided input should generally be done on render time, not storing

time.

Overview of required steps for implementing basic validation in your application:

1. Validate user input on the user interface
2. Customize AntiSamy to filter input on form submit
3. Do back-end validation and add messages to the session objects
4. Show validation feedback on the screen using liferay-ui tag library
5. Clean user-contributed output

Internationalizing the Application

Internationalization (i18n) means designing an application so that it isn't hardwired to one language only.

Internationalization is not just about localizing your application, but about a preferable design pattern for any user interface-centric applications. Even if you don't intend to have your application support multiple languages, internationalization lets you create a loose coupling between the display messages and the business logic, making the application more manageable.

Localization (i10n) is similar to internationalization, but adds support to a specific language.

About Language Keys

Language keys are unique string identifiers for the display messages. The same keys can be reused in the code as many times as needed. As a good practice, you should use describing keys like *submit-form* to improve code readability. Parametrization of keys is supported.

Language files

Language files, or localization files, contain a list of key value pairs for a single language. The default `Language.properties` file, which serves as a fallback default file, is automatically generated by a portlet module template and is located in the `src/main/resources/content` folder.

Language files have the following naming syntax:

`Language_LANGUAGE_CODE.properties`, where LANGUAGE_CODE is replaced either by:

- ISO 639-1 standard language code, or
- ISO 639-1 language code and ISO-3166 country code, separated by an underscore

Examples:

- `Language_en.properties` (all the English variants)
- `Language_en_US.properties` (US English)

| Language files should use UTF-8 encoding.

Configuring Language Resources

In order to use the language files, portlets and other components have to be made aware of the available resource bundles. In the case of portlets, this is done by defining the `javax.portlet.resource-bundle` component property. When using Liferay portlet module templates, default resources and component properties are created automatically.

Localization Example

Let's now have a look at a concrete localization example.

First, we have a portlet component and the `javax.portlet.resource-bundle` property. Notice that the value of the property is actually pointing to `src/main/resources/content/Language.properties`. The language file then contains a list of key value pairs for a specific language. Display messages are referenced by their respective keys, in this example, by using a Liferay tag library:

Portlet class

```
@Component(  
    immediate = true,  
    property = {  
        ...  
        "javax.portlet.resource-bundle=content.Language",  
        ...  
    })
```

```
    },
    service = Portlet.class
)
public class LocalizationExamplePortlet extends MVCPortlet {
    ...
}
```

Language.properties

```
hello-stranger=Hello stranger
hello-you=Hello you
my-favourite-dogs-are-x-and-x=My favourite-dogs are {0} and {1}
```

view.jsp

```
<liferay-ui:message key="hello-stranger" />
<liferay-ui:message key="my-favourite-dogs-are" arguments="<%=
new String[] {"Ryder", "Marshall"} %>"/>
```

When using tag libraries, notice that when a matching language key is not found in the portlet language file, the lookup falls back to the portal resource bundle. If the value is not found, the key is shown.

Note on Localizing Portlet Name

Portlet standard message localization follows a special pattern:

```
javax.portlet.<portlet-name-field>.<portlet-id>=<translation>
```

Where the *portlet-name-field* can be one of the following:

- description
- display-name
- keywords
- short-title
- title

Gradebook portlet display name:

```
javax.portlet.display-name.gradebook-web=Gradebook Portlet
```

Localization in the Back-End

Accessing the localization resources in a Java class requires loading the resource bundle manually. The `target` property of the `ResourceBundleLoader` reference is used to filter the resource:

Portlet class

```
public void doView(RenderRequest renderRequest, RenderResponse re  
nderResponse)  
throws IOException, PortletException {  
  
    String helloStranger = getResourceBundleLocalization("hello-s  
tranger", locale);  
    String myFavoriteDogsAre = getResourceBundleLocalization("my-  
favourite-dogs-are-x-and-x",  
        locale, "Ryder", "Marshall");  
}  
  
private String getLanguageLocalization(String key, Locale locale,  
Object...objects) {  
  
    ResourceBundle resourceBundle = _resourceBundleLoader.loadRes  
ourceBundle(locale);  
    String value = ResourceBundleUtil.getString(resourceBundle, k  
ey, objects);  
    return value==null ? _language.format(locale, key, objects) :  
    value;  
}  
  
@Reference(  
    target = "(bundle.symbolic.name=com.liferay.training.localiza  
tion.example)",  
    unbind = "-"  
)
```

```
private ResourceBundleLoader _resourceBundleLoader;  
  
@Reference  
private Language _language;
```

Sharing Resource Bundles Between Modules

In modular OSGi design, a single module is self-contained and usually contains all the resources it needs. In a multi-module application, however, you'll sometimes want either to centralize all localization resources in a dedicated module or let modules access localization resources from each other.

`-liferay-aggregate-resource-bundles` bnd instruction lets you use other resource bundles along with module's own resources:

```
-liferay-aggregate-resource-bundles: \  
[bundle.symbolic.name1], \  
[bundle.symbolic.name2]
```

The current module's resource bundle is prioritized over those of the listed modules.

Adding CSS Resources

Portlet CSS files are defined in portlet component properties. The following properties are available, each of them allowing multiple values:

- `com.liferay.portlet.header-portal-css`
- `com.liferay.portlet.header-portlet-css`
- `com.liferay.portlet.footer-portal-css`
- `com.liferay.portlet.footer-portlet-css`

By default, header and footer CSS files are injected respectively in the page header and footer. The difference between `portal-css` and `portlet-css` is the context path with which the former is set to portal and the latter to the portlet context. Portlet CSS files are typically placed in the `META-INF/resources/css` folder.

Liferay uses SASS compiler. Files suffixed with `.scss` are SASS-compiled automatically:

CSS file

```
css-portlet/src/main/resources/META-INF/resources/css/main.scss
```

Component properties

```
...
"com.liferay.portlet.header-portlet-css=/css/main.css",
...
```

Generated HTML

```
<head>
...
<link data-senna-track="temporary" href="http://localhost:8080/o/com.liferay.training.css/css/main.css?browserId=other&themeId=classic_WAR_classictheme&languageId=en_US&b=7100&t=1526032024000" id="605088bd" rel="stylesheet" type="text/css">
...
```

Notice that the reference to a CSS file in the portlet's properties is still `.css` even if you use the `.scss` as a suffix

Adding JavaScript Resources

Following the logic of portlet component CSS properties, portlet JavaScript files can be defined with:

- `com.liferay.portlet.header-portal-javascript`
- `com.liferay.portlet.header-portlet-javascript`
- `com.liferay.portlet.footer-portal-javascript`
- `com.liferay.portlet.footer-portlet-javascript`

The property values can be local paths or external URLs. If Javascript files are hosted locally, they are by convention stored in `META-INF/resources/js` folder. While this approach for including Javascripts works in simple cases, there are limitations to it especially when using 3rd party libraries. For example, scripts defined in the properties are loaded synchronously and they cannot be scoped or encapsulated. In more complex application it's recommended to use NPM based approaches. See the [Liferay Blade samples](#) for some options.

Portlet JavaScript resources are bundled and minified at compile time and served through combo servlet. An example:

main1.js

```
console.log("This is the main1.js file.")
```

main2.js

```
console.log("This is the main2.js file.")
```

Component Properties

```
@Component(  
    immediate = true,  
    property = {  
        ...  
        "com.liferay.portlet.header-portlet-javascript=/js/main1.  
js",  
        "com.liferay.portlet.header-portlet-javascript=/js/main2.  
js",  
        ...  
    },  
    service = Portlet.class  
)  
public class CssAndJavascriptExample extends MVCPortlet {  
}
```

Generated HTML

```
<head>
  ...
    <script data-senna-track="temporary" src="/combo?browserId=other&minifierType=&themeId=classic_WAR_classictheme&languageId=en_US&b=7100&CssAndJavascriptExample_INSTANCE_3sSKq0IS8KsX:%2Fjs%2Fmain1.js&CssAndJavascriptExample_INSTANCE_3sSKq0IS8KsX:%2Fjs%2Fmain2.js&t=1526034802000" type="text/javascript">&lt;/script>
  ...
</head>
```

The bundled, single JavaScript file contents:

```
console.log("This is the main1.js file.")
console.log("This is the main2.js file.")
```

Configuring JavaScript

To configure Javascript, look for the properties with the following prefixes in the [portal properties](#):

- javascript
- minifier
- combo

Liferay JavaScript API

The Liferay JavaScript object is available on all the pages and contains helpful platform utilities like:

- **Liferay.Browser**: browser capabilities detection
- **Liferay.ThemeDisplay**: ThemeDisplay object
- **Liferay.Service**: invokes Liferay services
- **Liferay.Language**: localization

An example of calling a Liferay service with `Liferay.Service` :

```
Liferay.Service(
```

```
'/user/get-user-by-email-address',
{
    companyId: Liferay.ThemeDisplay.getCompanyId(),
    emailAddress: 'test@liferay.com'
},
function(obj) {
    console.log(obj);
}
);
```

Good to Know

Alloy UI and jQuery libraries are globally available, but the platform doesn't restrict you from using any other preferred library. Also, support for ES2015 as well as NPM modules is available as well as a configurable AMD loader. These topics are discussed more in detail in the Front-End Developer course.

Links and Resources

- Alloy UI Validator: <https://github.com/liferay/alloy-ui/blob/master/src/aui-form-validator/js/aui-form-validator.js>
- Internationalizing Liferay applications: https://dev.liferay.com/en/develop/tutorials-/knowledge_base/7-2/internationalization
- Using JavaScript in Portlets: https://dev.liferay.com/develop/tutorials-/knowledge_base/7-2/using-javascript-in-your-portlets
- Liferay taglibs: <https://docs.liferay.com/portal/7.2-latest/taglibs/util-taglib/>

Knowledge Check

- In modular Liferay applications, the _____ is usually located in the _____ module.
- The web module typically contains both the _____ (user interface) and the _____ (portlet) layers.
- _____ are Liferay-provided, _____ meant to be used in conjunction with Liferay MVC portlets.
- There are three types of MVC commands:
 - _____
 - _____
 - _____

Exercises

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

Option 1: Use the Command Line Blade Tools

1. Open command line shell in your Liferay Workspace `modules/gradebook` folder.

2. **Run** command:

```
blade create -t mvc-portlet -p com.liferay.training.gradebook.  
web -c GradebookPortlet gradebook-web
```

3. **Run** Gradle refresh on the IDE.

Option 2: Use Developer Studio Wizard

1. **Launch** the *Liferay Module Project* wizard in Developers Studio.
2. **Use** the following information for the first step:
 - **Project Name:** "gradebook-web"
 - **Use Default Location:** uncheck and browse to *modules/gradebook* subfolder
 - **Build Type:** Gradle
 - **Liferay Version:** 7.2
 - **Project Template:** mvc-portlet
3. **Click Next**, and use the following information for the second step:
 - **Component Class Name:** "Gradebook"
 - **Package Name:** "com.liferay.training.gradebook.web"
4. **Click Finish** to close the wizard.

Declare Dependencies

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

1. Open the `build.gradle` of *gradebook-web* project.
2. Implement the new dependencies as follows:

```
compileOnly group: 'com.liferay', name: 'com.liferay.petra.function'  
compileOnly group: 'com.liferay', name: 'com.liferay.frontend.taglib.clay'  
compileOnly project(":modules:gradebook:gradebook-api")
```

Notice 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.

Set Portlet Properties

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:

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",
```

Set the Portlet Name

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):

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_portlet_GradebookPortlet";
```

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=Gradebook
```

```
javax.portlet.display-name.com_liferay_training_gradebook_web
_portlet_GradebookPortlet=Gradebook
javax.portlet.keywords.com_liferay_training_gradebook_web_por
tlet_GradebookPortlet=Gradebook
javax.portlet.short-title.com_liferay_training_gradebook_web_
portlet_GradebookPortlet=Gradebook
javax.portlet.title.com_liferay_training_gradebook_web_portle
t_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.fu
nction'

    compileOnly group: "com.liferay.portal", name: "com.liferay.p
ortal.kernel"
    compileOnly group: "com.liferay.portal", name: "com.liferay.u
til.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.compon
ent.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_portlet_GradebookPortlet";  
}
```

Language.properties

```
javax.portlet.description.com_liferay_training_gradebook_web_port  
let_GradebookPortlet=Gradebook  
javax.portlet.display-name.com_liferay_training_gradebook_web_por  
tlet_GradebookPortlet=Gradebook  
javax.portlet.keywords.com_liferay_training_gradebook_web_portlet  
_GradebookPortlet=Gradebook  
javax.portlet.short-title.com_liferay_training_gradebook_web_port  
let_GradebookPortlet=Gradebook  
javax.portlet.title.com_liferay_training_gradebook_web_portlet_Gr  
adebookPortlet=Gradebook  
gradebook.caption=Hello from Gradebook!
```

Deploy the Module

1. **Drag** the *gradebook-web* onto the Liferay server to deploy the module.
 - You should see the following message in the log:

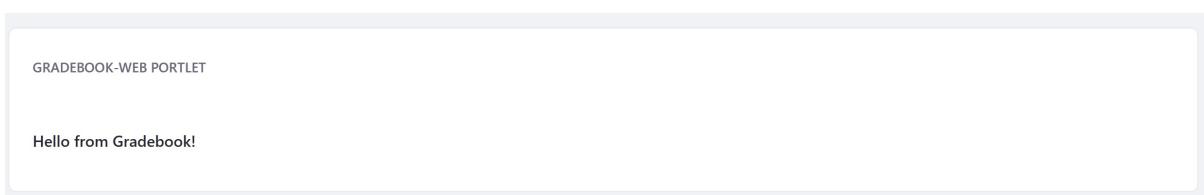
```
STARTED com.liferay.training.gradebook.web_1.0.0
```

Test the Module

Dev Studio's hot deploy feature allows us to see the changes on the user interface in almost real-time as we work with the code. Let's do a quick test to see how this feature works:

1. **Open** your browser to <http://localhost:8080> and sign in.
2. **Click** the *Add* icon on the top right corner of the page.
3. **Expand** the *category.training* category in the *Widgets* menu.
4. **Add** the *gradebook-web* portlet on the page.

Exercise: Create the Gradebook Web Module



Exercises

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

Implement the init.jsp

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.

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/t
ld/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"%>

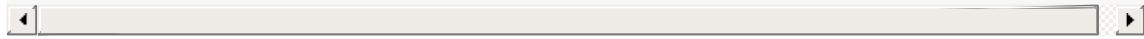
<%@ page import="com.liferay.portal.kernel.language.LanguageU
til"%>
<%@ page import="com.liferay.portal.kernel.portlet.LiferayWin
dowState"%>
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil"%>

<%@ page import="com.liferay.training.gradebook.model.Assignm
ent"%>
<%@ page import="com.liferay.training.gradebook.web.constants
.MVCCCommandNames"%>

<liferay-frontend:defineObjects />

<liferay-theme:defineObjects />

<portlet:defineObjects />
```



Implement the view.jsp

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.

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

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

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

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

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

    <clay:management-toolbar
        disabled="${assignmentCount eq 0}"
        displayContext="${assignmentsManagementToolbarDisplayCont
ext}"
        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="${assignmen
ts}" />

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

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

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

```

<%-- Iterator / Paging --%>

<liferay-ui:search-iterator
    displayStyle="${assignmentsManagementToolbarDisplayCo
ntext.getDisplayStyle()}"
    markupView="lexicon"
/>
</liferay-ui:search-container>
</div>
```

```

We generally rely heavily on `liferay-ui` and `clay` tag libraries to simplify the JSP code. Take a look at the [Search container](#)) component, which is one of the most common Liferay UI components, facilitating the task of showing item lists and search results.

## Implement the Other JSP Files

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

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="<%=%MVCC
ommandNames.VIEW_ASSIGNMENT %>" />
 <portlet:param name="redirect" value="${currentURL}" />
 <portlet:param name="assignmentId" value="${entry.assignm
entId}" />
</portlet:renderURL>

<c:choose>

```

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

<c:when
 test='${assignmentsManagementToolbarDisplayContext.getDisplayStyle().equals("descriptive")}'>

 <%-- 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()
 , true);
 %>

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

 <h4>
 <aui:a href="${viewAssignmentURL}">
 ${entry.getTitle(locale)}
 </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().equals("icon")}'>

 <%
 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.getTitle(locale)}"
 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.getDescription(locale)) %>
 </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(locale) %>">
 />

 <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="deleteAssignmentURL">
 <portlet:param name="redirect" value="${currentURL}" />
 </portlet:actionURL>

```

```
<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.getTitle(locale)}</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.getDescription(locale)}</dd>
 </dl>
 </div>
</div>
```

## Implement the MVC Render Command for Showing the Assignments List

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.

### Implement MVCCCommandNames.java

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.

1. **Create** a class

```
com.liferay.training.gradebook.web.constants.MVCCCommandNames .
```

2. **Implement** as follows:

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

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

 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";
}
```

Now implement the render command for showing the assignments list:

1. **Create** a class

```
com.liferay.training.gradebook.web.portlet.action.ViewAssignment
sMVCRenderCommand .
```

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.MVCCmdNames;
import com.liferay.training.gradebook.web.display.context.AssignmentsManagementToolbarDisplayContext;

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=" + MVCCmdNames.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 assignment 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.getAssignmentsCountByKeywords(
 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)
;

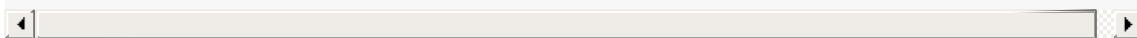
 AssignmentsManagementToolbarDisplayContext assignmentManagementToolbarDisplayContext =
 new AssignmentsManagementToolbarDisplayContext(
 liferayPortletRequest, liferayPortletResponse
,
 _portal.getHttpServletRequest(renderRequest))
;

 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=" + MVCCCommandNames.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.get
SimpleDateFormat(
 "EEEEEE, MMMMM dd, yyyy", renderRequest.getLoc
ale());

 // Set attributes to the request.

 renderRequest.setAttribute("assignment", assignme
nt);
 renderRequest.setAttribute(
 "dueDate", dateFormat.format(assignment.getDu
eDate()));
 renderRequest.setAttribute(
 "createDate", dateFormat.format(assignment.ge
tCreateDate()));

 // Set back icon visible.

 PortletDisplay portletDisplay = themeDisplay.getP
ortletDisplay();

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

 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;
}
```

## Implement the Back-End Class for the UI Management Toolbar

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 documentation [here](#):

1. **Create** a class

```
com.liferay.training.gradebook.web.display.context.AssignmentsMa
nagementToolbarDisplayContext .
```

2. **Implement** as follows:

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

import com.liferay.frontend.taglib.clay.servlet.taglib.display.co
ntext.BaseManagementToolbarDisplayContext;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.Creat
ionMenu;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.Dropd
ownItem;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.Dropd
ownItemList;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.ViewT
ypeItem;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.ViewT
ypeItemList;
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.PortletPreferencesFactor
yUtil;
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.GradebookPort
letKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNam
es;

import java.util.List;

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

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

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

 super(
```

```

 liferayPortletRequest, liferayPortletResponse, httpSe
rvletRequest);

 _portalPreferences = PortletPreferencesFactoryUtil.getPor
talPreferences(
 liferayPortletRequest);

 _themeDisplay = (ThemeDisplay)httpServletRequest.getAttri
bute(
 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.createRenderUR
L(),
 "mvcRenderCommandName", MVCCommandNam
es.EDIT_ASSIGNMENT,
 "redirect", currentURLObj.toString())
;
 dropdownItem.setLabel(
 LanguageUtil.get(request, "add-assig
ment"));
 });
 }
 };
 }
}

```

```
@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-style",
 "descriptive");
 }
 else {
 _portalPreferences.setValue(
 GradebookPortletKeys.Gradebook, "assignments-display-style",
 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.createRende
rURL();

 searchURL.setProperty(
 "mvcRenderCommandName", MVCCCommandNames.VIEW_ASSIGNME
NTS);

 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.createRend
erURL();

 portletURL.setParameter(
 "mvcRenderCommandName", MVCCmdNames.VIEW_ASSI
GNMENTS);

 int delta =
 ParamUtil.getInteger(request, SearchContainer.DEF
AULT_DELTA_PARAM);

 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
OrderByCol()));
 dropdownItem.setHref(
 _getCurrentSortingURL(), "orderByCol"
, "title");
 dropdownItem.setLabel(
 LanguageUtil.get(request, "title"));
 });
 add(
 dropdownItem -> {
 dropdownItem.setActive(

```

```
 "createDate".equals(getOrderByCol())))
 ;
 dropdownItem.setHref(
 _getCurrentSortingURL(), "orderByCol"
 ,
 "createDate");
 dropdownItem.setLabel(
 LanguageUtil.get(request, "create-dat
e"));
 });
}
};

/**
 * Returns the current sorting URL.
 *
 * @return current sorting portlet URL
 *
 * @throws PortletException
 */
private PortletURL _getCurrentSortingURL() throws PortletExce
ption {
 PortletURL sortingURL = PortletURLUtil.clone(
 currentURLObj, liferayPortletResponse);

 sortingURL.setParameter(
 "mvcRenderCommandName", MVCCCommandNames.VIEW_ASSIGNME
NTS);

 // Reset current page.

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

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

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

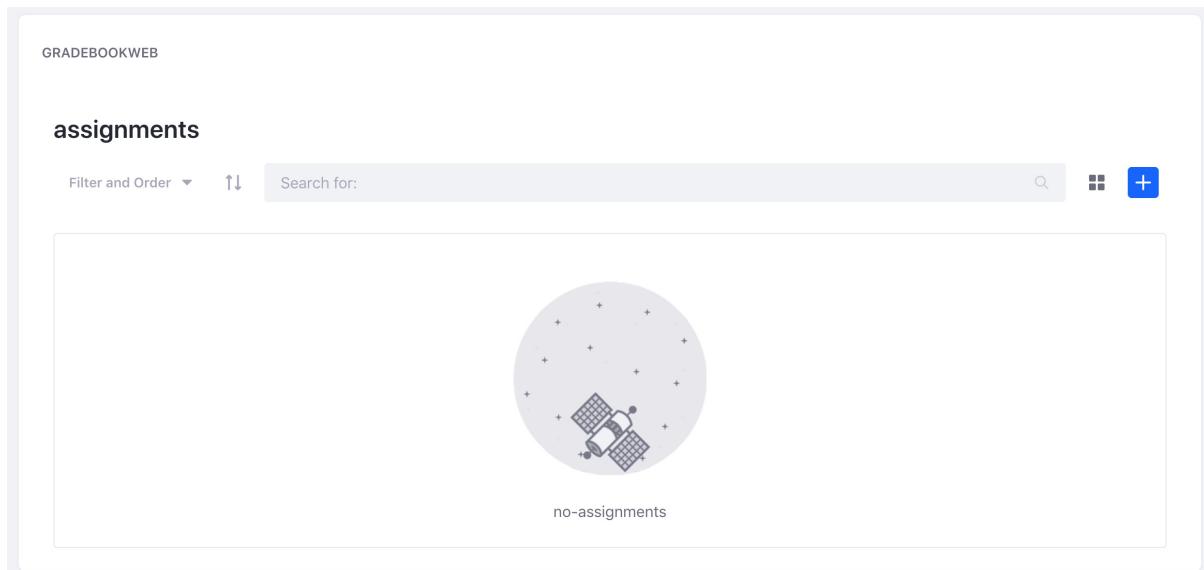
```
 }

 return sortingURL;
}

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

## Test the User Interface

After the module has restarted successfully, refresh the browser to see the changes:



## Exercises

# 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

We'll use the AUI tag library to create the Assignment editing form:

1. **Create** a JSP file `src/main/resources/META-INF/resources/assignment/edit_assignment.jsp` .
2. **Implement** as follows:

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

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

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

```
<c:set var="editTitle" value="edit-assignment"/>
</c:when>
<c:otherwise>
 <portlet:actionURL var="assignmentActionURL" name="<%>
=MVCCCommandNames.ADD_ASSIGNMENT %>">
 <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>
 </aui:fieldset>
 </aui:fieldset-group>
 </aui:form>
</div>
```

```
<%-- 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.redirect}" type="cancel" />
</aui:button-row>
</aui:form>

</div>
```

## Implement an MVC Render Command for Switching to the Editing View

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 `AssignmentsManagementToolbarDisplayContext.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", MVCCCommandNames.E
DIT_ASSIGNMENT,
 "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 yet registered for the command name `MVCCCommandNames.EDIT_ASSIGNMENT` :

1. **Create** a class

```
com.liferay.training.gradebook.web.portlet.action.EditAssignment
MVCRenderCommand .
```

2. **Implement** as follows (notice the return value of the `render()` method):

```
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.MVCComan
dNames;

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.Gradeboo
k,
 "mvc.command.name=" + MVCComanNames.EDIT_ASSIGNMENT
 },
 service = MVCRenderCommand.class
)
public class EditAssignmentMVCRenderCommand implements MVCRen
derCommand {

 @Override
 public String render(
 RenderRequest renderRequest, RenderResponse renderRes
ponse)
 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
(assignmentId);
 }
 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.getPortl
etDisplay();

portletDisplay.setShowBackIcon(true);

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

portletDisplay.setURLBack(redirect);

// Set assignment to the request attributes.

renderRequest.setAttribute("assignment", assignment);
renderRequest.setAttribute("assignmentClass", Assignm
```

```
ent.class);

 return "/assignment/edit_assignment.jsp";
 }

 @Reference
 private AssignmentService _assignmentService;

}
```

Test and click the plus button. 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.

[◀ GRADEBOOK](#)

**add-assignment**

Title \*

Description \*

Due Date \*

04/04/2019 12:11 PM

Save Cancel

## Implement an MVC Action Command for Adding an Assignment

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:

1. **Create** a class

```
com.liferay.training.gradebook.web.portlet.action.AddAssignmentM
VCActionCommand .
```

2. **Implement** 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.Gradeboo
k,
 "mvc.command.name=" + MVCCCommandNames.ADD_ASSIGNMENT
},
service = MVCActionCommand.class
)
public class AddAssignmentMVCActionCommand extends BaseMVCAct
ionCommand {

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

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

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

 // Get parameters from the request.

 // Use LocalizationUtil to get a localized parameter.

 Map<Locale, String> titleMap =
 LocalizationUtil.getLocalizationMap(actionRequest
, "title");

 Map<Locale, String> descriptionMap =
 LocalizationUtil.getLocalizationMap(actionRequest
, "description");

 Date dueDate = ParamUtil.getDate(
 actionRequest, "dueDate",
 DateFormatFactoryUtil.getDate(actionRequest.getLo
```

```
cale()));

try {

 // Call the service to add a new assignment.

 _assignmentService.addAssignment(
 themeDisplay.getScopeGroupId(), titleMap, des
criptionMap, 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 Editing an Assignment

1. **Create** a class

```
com.liferay.training.gradebook.web.portlet.action.EditAssignment
MVCActionCommand .
```

2. **Implement** 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.BaseMVCA
ctionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActio
nCommand;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.ServiceContextFactor
y;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.LocalizationUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.training.gradebook.exception.AssignmentVal
idationException;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentServ
ice;
import com.liferay.training.gradebook.web.constants.Gradebook
PortletKeys;
import com.liferay.training.gradebook.web.constants.MVCComan
dNames;

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 editing assignments.
*
* @author liferay
*
*/

@Component(
 immediate = true,
 property = {
 "javax.portlet.name=" + GradebookPortletKeys.Gradeboo
k,
 "mvc.command.name=" + MVCCCommandNames.EDIT_ASSIGNMENT
 },
 service = MVCActionCommand.class
)
public class EditAssignmentMVCActionCommand extends BaseMVCActionCommand {

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

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

 // Get parameters from the request.

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

 Map<Locale, String> titleMap =
 LocalizationUtil.getLocalizationMap(actionRequest
, "title");

 Map<Locale, String> descriptionMap = LocalizationUtil
 .getLocalizationMap(actionRequest, "description");

 Date dueDate = ParamUtil.getDate(
```

```
 actionRequest, "dueDate",
 DateFormatFactoryUtil.getDate(actionRequest.getLo
cale()));

 try {

 // Call the service to update the assignment

 _assignmentService.updateAssignment(
 assignmentId, titleMap, descriptionMap, dueDa
te, serviceContext);

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

 ave.printStackTrace();

 actionResponse.setRenderParameter(
 "mvcRenderCommandName", MVCCCommandNames.EDIT_
ASSIGNMENT);

 }
 catch (PortalException pe) {

 pe.printStackTrace();

 actionResponse.setRenderParameter(
 "mvcRenderCommandName", MVCCCommandNames.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** 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.util.ParamUtil;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCmdNames;

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=" + MVCCmdNames.DELETE_ASSIGNMENT
 },
}
```

```
 service = MVCActionCommand.class
)
public class DeleteAssignmentMVCActionCommand extends BaseMVC
ActionCommand {

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

 // Get assignment id from request.

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

 try {

 // Call service to delete the assignment.

 _assignmentService.deleteAssignment(assignmentId)
;

 }

 catch (PortalException pe) {
 pe.printStackTrace();
 }

 }

 @Reference
 protected AssignmentService _assignmentService;
}
```

## Test the Application

After the module has restarted successfully, refresh the browser window and try adding, editing, viewing, deleting Assignments:

GRADEBOOK

**assignments**

Filter and Order ▾    ↑↓    Search for:           

My First Assignment	⋮
 	⋮
Is this	⋮

## Exercises

# 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

### Customize the `AssignmentValidationException` to Support Message Stacking

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`.

1. Open the

```
com.liferay.training.gradebook.exception.AssignmentValidationException.java
```

in the *gradebook-api* module.

2. Implement the following 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.

## Implement an Assignment Validator Interface

1. **Go to** the *gradebook-api* module.

2. **Create** an interface

```
com.liferay.training.gradebook.validator.AssignmentValidator .
```

3. **Implement** as follows:

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

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

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

public interface AssignmentValidator {
```

```
/**
```

```
* Validates an Assignment
*
* @param titleMap
* @param descriptionMap
* @param dueDate
* @throws AssignmentValidationException
*/
public void validate(
 Map<Locale, String> titleMap, Map<Locale, String> descriptionMap, 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:\n
 com.liferay.training.gradebook.exception,\n
 com.liferay.training.gradebook.model,\n
 com.liferay.training.gradebook.service,\n
 com.liferay.training.gradebook.service.persistence,\n
 com.liferay.training.gradebook.validator\n
-check: EXPORTS\n
-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.AssignmentValidator
rImpl .
```

2. **Implement** as follows:

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

import com.liferay.portal.kernel.util.LocaleUtil;
import com.liferay.portal.kernel.util.MapUtil;
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 java.util.Locale;
import java.util.Map;

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

@Component(
 immediate = true,
 service = AssignmentValidator.class
)
public class AssignmentValidatorImpl implements AssignmentValidator {

 /**
 * Validates assignment values and throws
 * {AssignmentValidationException} if the assignment values are not
 * valid.
 *
 * @param titleMap
 * @param descriptionMap
 * @param dueDate
 * @throws AssignmentValidationException
 */
 public void validate(
 Map<Locale, String> titleMap, Map<Locale, String> descriptionMap, Date dueDate)
 throws AssignmentValidationException {
```

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

 if (!isValidAssignment(titleMap, descriptionMap, dueDate, errors)) {
 throw new AssignmentValidationException(errors);
 }
 }

 /**
 * Returns <code>true</code> if the given fields are valid for an
 * assignment.
 *
 * @param titleMap
 * @param descriptionMap
 * @param dueDate
 * @param errors
 * @return boolean <code>true</code> if assignment is valid, otherwise
 * <code>false</code>
 */
 private boolean isValidAssignment(
 final Map<Locale, String> titleMap, final Map<Locale,
 String> descriptionMap,
 final Date dueDate, final List<String> errors) {

 boolean result = true;

 result &= isTitleValid(titleMap, errors);
 result &= isDueDateValid(dueDate, errors);
 result &= isDescriptionValid(descriptionMap, 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 lis
```

```
t.
*
* @param descriptionMap
* @param errors
* @return boolean <code>true</code> if description is va
lid, otherwise
* <code>false</code>
*/
private boolean isDescriptionValid(
 final Map<Locale, String> descriptionMap, final List<
String> errors) {

 boolean result = true;

 // Verify the map has something

 if (MapUtil.isEmpty(descriptionMap)) {
 errors.add("assignmentDescriptionEmpty");
 result = false;
 }
 else {

 // Get the default locale

 Locale defaultLocale = LocaleUtil.getSiteDefault(
);

 if ((Validator.isBlank(descriptionMap.get(default
Locale)))) {
 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 need it.
 *
 * @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 titleMap
 * @param errors
 * @return boolean <code>true</code> if the title is valid, otherwise
 * <code>false</code>
 */
private boolean isTitleValid(

```

```

 final Map<Locale, String> titleMap, final List<String> errors) {

 boolean result = true;

 // Verify the map has something

 if (MapUtil.isEmpty(titleMap)) {
 errors.add("assignmentTitleEmpty");
 result = false;
 }
 else {

 // Get the default locale.

 Locale defaultLocale = LocaleUtil.getSiteDefault();
 }

 if ((Validator.isBlank(titleMap.get(defaultLocale)))) {
 errors.add("assignmentTitleEmpty");
 result = false;
 }
}

return result;
}
}

```

## Implement Validation in the Assignment Service

The Assignment service can not only be accessed from the portlet user interface, but, for example, through a JSON web service call or even from a completely different application. To ensure validity, we have to implement validation in the service layer addAssignment() and updateAssignment() methods.

- 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, Map<Locale, String> titleMap, Map<Locale, S
tring> descriptionMap,
 Date dueDate, ServiceContext serviceContext)
throws PortalException {

 // Validate assignment parameters.

 _assignmentValidator.validate(titleMap, descriptionMap, d
ueDate);
```

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

```
public Assignment updateAssignment(
 long assignmentId, Map<Locale, String> titleMap, Map<Loca
le, String> descriptionMap,
 Date dueDate, ServiceContext serviceContext)
throws PortalException {

 // Validate assignment parameters.

 _assignmentValidator.validate(titleMap, descriptionMap, d
ueDate);
```

6. **Organize** missing imports.

7. **Rebuild** the service.

## Implement Feedback Messages Dispatching on the Controller Layer

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.

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=f
alse"
```

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:

1. **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.BaseMVCActio
nCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCom
mand;
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)
 {
 Assignment assignment = actionRequest.getParameterMap()
 .get("assignment");
 ...
 }
}
```

```
throws Exception {

 ThemeDisplay themeDisplay =
 (ThemeDisplay) actionRequest.getAttribute(WebKeys.THE
ME_DISPLAY);

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

 // Get parameters from the request.

 // Use LocalizationUtil to get a localized parameter.

 Map<Locale, String> titleMap =
 LocalizationUtil.getLocalizationMap(actionRequest, "t
itle");

 Map<Locale, String> descriptionMap =
 LocalizationUtil.getLocalizationMap(actionRequest, "d
escription");

 Date dueDate = ParamUtil.getDate(
 actionRequest, "dueDate",
 DateFormatFactoryUtil.getDate(actionRequest.getLocale
()));

try {

 // Call the service to add a new assignment.

 _assignmentService.addAssignment(
 themeDisplay.getScopeGroupId(), titleMap, descrip
tionMap, dueDate,
 serviceContext);

 // 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, key));

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

 }
 catch (PortalException pe) {

 // Set error messages from the service layer.

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

 actionResponse.setRenderParameter(
 "mvcRenderCommandName", MVCCCommandNames.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.BaseMVCAction

```

```
nCommand;
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.LocalizationUtil;
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 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 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 BaseMVCAction
Command {

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

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

 // Get parameters from the request.

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

 Map<Locale, String> titleMap =
 LocalizationUtil.getLocalizationMap(actionRequest, "t
itle");

 Map<Locale, String> descriptionMap =
 LocalizationUtil.getLocalizationMap(actionRequest, "d
escription");

 Date dueDate = ParamUtil.getDate(
 actionRequest, "dueDate",
 DateFormatFactoryUtil.getDate(actionRequest.getLocale
()));

 try {

 // Call the service to update the assignment

 _assignmentService.updateAssignment(
 assignmentId, titleMap, descriptionMap, 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, key));

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

catch (PortalException pe) {

 // Get error messages from the service layer.

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

 actionResponse.setRenderParameter(
 "mvcRenderCommandName", MVCCCommandNames.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 actionRespons
e)
throws Exception {

 // Get assignment id from request.

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

 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;
}
```



## Display Feedback Messages on the User Interface

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.

## init.jsp

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

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

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

## view.jsp

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

1. **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(life
rayPortletRequest, "serviceErrorDetails") %>' key="error.assign
ment-service-error" />
</liferay-ui:error>
<liferay-ui:success key="assignmentAdded" message="assignment
-added-successfully" />
<liferay-ui:success key="assignmentUpdated" message="assignme
nt-updated-successfully" />
<liferay-ui:success key="assignmentDeleted" message="assignme
nt-deleted-successfully" />
```

## edit\_assignment.jsp

1. Add `<liferay-ui>` tags to `src/main/resources/META-INF/resources/assignment/edit_assignment.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='<%= SessionErrors.get(liferayPortletRequest, "serviceErrorDetails") %>' />
</liferay-ui:error>
<liferay-ui:error key="assignmentTitleEmpty" message="error.assignment-title-empty" />
<liferay-ui:error key="assignmentDescriptionEmpty" message="error.assignment-description-empty" />
```

## Test the Server-Side Validation

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

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:

The screenshot shows a web page titled 'add-assignment'. At the top, there is a red error message box containing the text 'Error: error.assignment-title-empty'. Below this, the form fields are displayed: 'Title' (empty), 'Description \*' (containing 'Description'), and 'Due Date \*' (set to '04/04/2019 07:37 PM'). At the bottom of the form are two buttons: 'Save' (highlighted in blue) and 'Cancel'.

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

## Implement Client-Side Validation

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_assignment.jsp`. Notice the already existing tag for the *description* fields:

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

Add two validators for the *title* field. One is for setting the field mandatory, and the other one checks valid characters:

1. **Open** the `edit_assignment.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_assignment.jsp` file will look like this:

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

<%-- Error messages. --%>
```

```

<liferay-ui:error key="serviceErrorDetails">
 <liferay-ui:message key="error.assignment-service-error" arguments='<%= SessionErrors.get(liferayPortletRequest, "serviceError Details") %>' />
</liferay-ui:error>
<liferay-ui:error key="assignmentTitleEmpty" message="error.assignment-title-empty" />
<liferay-ui:error key="assignmentDescriptionEmpty" message="error.assignment-description-empty" />

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

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

 <c:set var="editTitle" value="edit-assignment"/>
 </c:when>
 <c:otherwise>
 <portlet:actionURL var="assignmentActionURL" name="<%=>MVC CommandNames.ADD_ASSIGNMENT %>">
 <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
-title-format" name="custom">
 function(val, fieldNode, ruleValue) {
 var wordExpression =
 new RegExp("^[^\\[\\]\\^$]*$");
 }
 </aui:validator>
 </aui:input>

 <%-- Description field. --%>

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

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

 <aui:input name="dueDate" />
 </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.redirect}" type="cancel" />
</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:

[◀ GRADEBOOK](#)

### add-assignment

The screenshot shows a form titled "add-assignment". It has three fields: "Title \*", "Description \*", and "Due Date \*". The "Title" field contains "\$dollars" and has a red error message below it: "error.assignment-title-format". The "Description" field is empty and has a red error message: "This field is required.". The "Due Date" field shows "04/04/2019" and "07:05 PM". At the bottom are "Save" and "Cancel" buttons.

Title \*

\$dollars

error.assignment-title-format

Description \*

Description  
This field is required.

Due Date \*

04/04/2019 07:05 PM

Save Cancel

## Exercises

### Add Localization Resources

#### Exercise Goals

- Review portlet component properties
- Change the localization file encoding
- Provide localization resources
- Test the user interface

#### Review Portlet Component Properties

Take a look at the component properties of `GradebookPortlet.java`. See the resource bundle property:

```
@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.init-param.add-process-action-success-acti
on=false",
 "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 {
}
```

The value of the property translates to a file system path

`src/main/resources/content/Language.properties`, which was done automatically by the module template. By setting this property, localization resources of `Language.properties` are available for Liferay JSP tag libraries automatically. To add support for another language, let's say for German, just add a new language properties file `src/main/resources/content/Language_de_DE.properties`.

## Change the Localization File Encoding

Default encoding for the language text files is ISO-8859-1. While it usually works for English, localizations for other languages with non-standard ASCII special characters will break.

Let's change the file encoding to UTF-8:

1. **Right-click** on the `src/main/resources/content/Language.properties` file in the *gradebook-web*.
2. **Click** on *Properties*.
3. **Change** the *Text file encoding* to UTF-8 and close the dialog.

## Provide Localization Resources

1. **Open** the `gradebook-web/src/main/resources/content/Language.properties` file.
2. **Implement** the file as follows:

```

This is the default localization file containing the key to dis-
play messages mappings.

Standard portlet display messages

javax.portlet.description.com_liferay_training_gradebook_web_port-
let_GradebookPortlet=Gradebook
javax.portlet.display-name.com_liferay_training_gradebook_web_port-
let_GradebookPortlet=Gradebook
```

```
javax.portlet.keywords.com_liferay_training_gradebook_web_portlet
 _GradebookPortlet=Gradebook
javax.portlet.short-title.com_liferay_training_gradebook_web_port
let_GradebookPortlet=Gradebook
javax.portlet.title.com_liferay_training_gradebook_web_portlet_Gr
adebookPortlet=Gradebook
#
Application category
#
category.training=Liferay Training

#
Asset name localization (Asset Publisher, search and permission
s UI)
#
model.resource.com.liferay.training.gradebook.model.Assignment=As
signment
model.resource.com.liferay.training.gradebook.model=Gradebook

#
Application Display Template localization
#
application-display-template-type=Gradebook template

#
Permission localizations
#
action.ADD_ASSIGNMENT=Add Assignment
action.ADD_SUBMISSION=Add Submission
action.DELETE_SUBMISSION=Delete Submission
action.EDIT_SUBMISSION>Edit Submission
action.GRADE_SUBMISSION=Grade Submissions
action.VIEW_SUBMISSIONS=View Submissions

#
Other messages
#
add-assignment=Add New Assignment
add-submission=Add New Submission
add-submission-for-x=Add New Submission for {0}
```

```
are-you-sure-to-delete=Are you sure to delete this?
assignment-added-successfully=Assignment added successfully
assignment-deleted-successfully=Assignment was deleted successfully
assignment-duedate=Assignment Due Date
assignment-information=Assignment Information
assignment-updated-successfully=Assignment updated successfully
assignments=Assignments
assignments-help-text=Please click the plus sign to add a new assignments.
edit-assignment>Edit
edit-submission-for-x>Edit Submission for {0}
error.assignment-date-empty=Due date cannot be empty.
error.assignment-description-empty=Description cannot be empty.
error.assignment-service-error=Service request failed with message: {0}
error.assignment-title-empty=Please enter a valid title.
error.only-one-submission-allowed=Only one submission per student is allowed.
error.assignment-title-format=Please enter letters, words, numbers, or standard punctuation.
error.submission-is-too-late>Your submission is too late.
error.submission-service-error=There was a problem with your submission.
error.submission-text-null=Submission text cannot be empty.
error.submission-text-too-short=Submission text too short.
error.submission-text-too-long=Submission text too long.
grade=Grade
gradebook-example-adt=Gradebook Application Display Template example
gradebook-portlet-instance-configuration-name=Gradebook Portlet Configuration
gradebook-service-configuration-name=Gradebook Service Configuration
grade-submission=Grade
grading=Grading
no-assignments>No assignments yet
no-comment>No comments yet.
no-submissions>No submissions for this assignment yet
not-graded=Not graded yet.
student=Student
```

```
submission-comment=Submission Comment
submission-added-successfully=Submission was created successfully
submission-deleted-successfully=Submission was deleted successfully
submission-graded-successfully=Submission was graded successfully
submission-information=Submission Information
submission-not-graded=Not graded
submission-settings=Submission Settings
submission-text=Submission Text
submissions=Submissions
submit-date=Submitted
this-is-an-example-adt=This is an Example ADT for the Gradebook
viewing-submissions-not-allowed=You don't have permissions to view submissions.
your-submission=Your Submission
```

Notice that some of the keys are used at later and optional exercise steps.

## Test the User Interface

1. Open the Gradebook application in your web browser and refresh the page. You should now be able to see the labels and messages correctly:

The screenshot shows a web-based Gradebook application. At the top left, it says "GRADEBOOK". Below that is a header with "Assignments" and a search bar labeled "Search for:". To the right of the search bar are icons for filtering, ordering, and adding new items. The main content area displays three assignment cards. Each card has a grid icon at the top, followed by a small circular profile picture on the left and a purple circular badge with "LD" on the right. The first card is titled "My First Assignment" with the subtitle "This is it.". The second card is titled "My second Assignment" with the subtitle "Is this". The third card is titled "My Third Assignment" with the subtitle "Is this." Each card also has a vertical ellipsis icon on the right side.

# Exercises

## Add CSS Resources

### Exercise Goals

- Add CSS resources
- Configure the portlet component to use the provided CSS resources
- Test the changes

The styling of the assignment list needs polishing. In the *Table* view, the author column is not aligned and the links should be underlined.

GRADEBOOK

#### Assignments

Title	Author	Create Date	Actions
My Third Assignment		7 Hours Ago	⋮
My second Assignment		7 Hours Ago	⋮
My First Assignment	LD Liferay Demo	7 Hours Ago	⋮

Provide CSS resources for the Gradebook portlet to fix the issues.

## Add CSS Resources

Let's first create a CSS file to provide our custom styles for the *gradebook-web* module.

1. **Create** a folder file `resources/META-INF/resources/css` .

2. **Create** a file `resources/META-INF/resources/css/main.scss` and implement as follows:

```
.gradebook-portlet {

 h1 {
 font-size: 1.7rem;
 margin: 20px 0 10px 0;
 }

 h2 {
 margin: 30px 0 10px 0;
 }

 .lfr-search-container-wrapper {
 a {
 text-decoration: underline;
 }

 .user-icon {
 float: left;
 }

 .user-details {
 vertical-align: sub;

 .user-name {
 color: inherit;
 }
 }
 }

 .submission-text {
 border: 1px solid #eee;
 border-radius: 5px;
 padding: 20px;
 }

 .assignment-metadata,
 .submission-metadata {
```

```
font-size: .9em;

dt {
 margin-top: 15px;
}

dd {

}

.edit-assignment {

 .assignment-description {
 font-size: .875rem;
 font-weight: 600;

 .reference-mark {
 font-size: 6px;
 }
 }
}
```

## Configure the Portlet Component

The portlet component needs to know where to load the CSS resources from. Also, it's a good practice to encapsulate portlet styles by wrapping the portlet in a CSS class.

1. Open the `GradebookPortlet.java` portlet class.
2. Add the following component properties:

```
"com.liferay.portlet.css-class-wrapper=gradebook-portlet",
"com.liferay.portlet.header-portlet-css=/css/main.css",
```

## Test the Changes

1. Refresh the browser to see the changes after the module has redeployed correctly.
2. Switch the list to the *Table* view using the button on the left side of the search bar, if

necessary. The Author column is now better aligned and the links have underlining:

GRADEBOOK

### Assignments

Filter and Order		Search for:	Actions
Title	Author	Create Date	
<a href="#">My Third Assignment</a>		7 Hours Ago	⋮
<a href="#">My second Assignment</a>		7 Hours Ago	⋮
<a href="#">My First Assignment</a>	LD	Liferay Demo	7 Hours Ago

# Implement Access Control

Liferay has a robust security model that allows for the configuration of fine-grained access control. Liferay's access control system lets you define who can use an application and who is allowed to add and edit a model resource.

For example, all applications dealing with user and site management are only accessible for Liferay Administrators and not for regular users. Wiki Pages can be created and edited by any member of a site, while Blogs posts by default can only be edited by their original author.

Access control in Liferay is based on four key concepts: Resources, Actions, Permissions, and Roles.

## Resources

A resource in Liferay's permission framework is a generic representation of any application or model entity that can be used as an action target. There are two distinct kinds of resources: portlet and model resources.

## Portlet Resources

A portlet resource represents a portlet application, like the Blogs or Wiki, and is identified in the permission system by a portlet's ID, defined by a `javax.portlet.name` property. By convention, a component's fully qualified classname with the dots replaced by underscores is used as the ID, as seen in the code snippet taken from Liferay's [Blogs Portlet](#):

### BlogsPortlet.java

```
@Component(
 immediate = true,
 property = {
 ...
 "javax.portlet.name=" + BlogsPortletKeys.BLOGS,
 ...
 },
 service = Portlet.class
```

```
)
public class BlogsPortlet extends BaseBlogsPortlet {
...
}
```

### BlogsPortletKeys.java

```
public class BlogsPortletKeys {

 public static final String BLOGS =
 "com_liferay_blogs_web_portlet_BlogsPortlet";

 ...
}
```

A best practice is to define the portlet's ID as a constant in a dedicated keys class in order to avoid ambiguities and misspellings.

## Model Resources

A Model Resource represents a model entity, usually managed by a portlet application. Examples of model resources include:

- A wiki page
- A document
- A site

Model resources are usually identified and referenced by an entity's fully qualified class name, for example:

- **Web Content:** com.liferay.journal.model.JournalArticle
- **BlogsEntry:** com.liferay.blogs.model.BlogsEntry

## Actions

Actions are operations that can be performed on a given resource, either on a portlet or a model resource.

Examples for portlet resource actions:

- **ADD\_TO\_PAGE**: gives users the ability to add a portlet to a page
- **CONFIGURATION**: lets users access a portlet's configuration menu
- **VIEW**: lets users view the portlet

Examples for model resource actions:

- ADD\_ENTRY
- UPDATE
- DELETE
- PERMISSIONS
- VIEW

The above example actions are taken from Liferay's core applications, but you can define custom actions for your own applications.

## Action Types

There are two types of actions:

1. **Top-level actions**: general model actions that can't be applied to an existing resource, for example, `ADD_ENTRY` .
2. **Resource actions**: actions applied to an existing resource, for example, `DELETE` .

By convention, the *top-level actions* are referenced by the package name of the respective service, for example, `com.liferay.blogs` , and the *resource actions* by the fully qualified name of the targeted model entity, for example, `com.liferay.blogs.model.BlogsEntry` .

## Permissions

A permission is when a *resource* meets an *action* or, more formally: a permission is an *action* that can be performed on a *resource*. The permission to update a Blogs post, for example, is defined by the combination of the BlogsEntry's model resource identifier and the UPDATE action: `com.liferay.blogs.model.BlogsEntry` + `UPDATE` .

## Permission Scopes

The scope of a permission defines how broadly the permission applies to the resources in the portal. There are four options:

1. **Company:** grants a user permissions for every resource of the type within the portal instance
2. **Group:** gives users permissions for every resource within a specified group
3. **Group Template:** similar to *Group* scope, except that it does not automatically apply to a specific group. A user must be a member of a group (generally either a site or an organization), and they must have been given the role within that group before they are granted its permissions.
4. **Individual:** applies only to a single resource of the type

## Roles

Roles are entities that bind all the previously mentioned concepts to a user or a user group. In other words, **roles are collections of permissions** that give users access to different parts of the platform and lets them perform certain tasks.

Roles can be assigned to:

- Users
- Sites
- Organizations
- User groups

Roles are always defined in one of the following scopes:

- Global
- Site or organization
- Within a site or organization (Team)

Roles defined for the *Global* scope apply to the entire portal instance. The most prominent global role is the (portal) Administrator role, which gives all its members control over every resource of the portal instance.

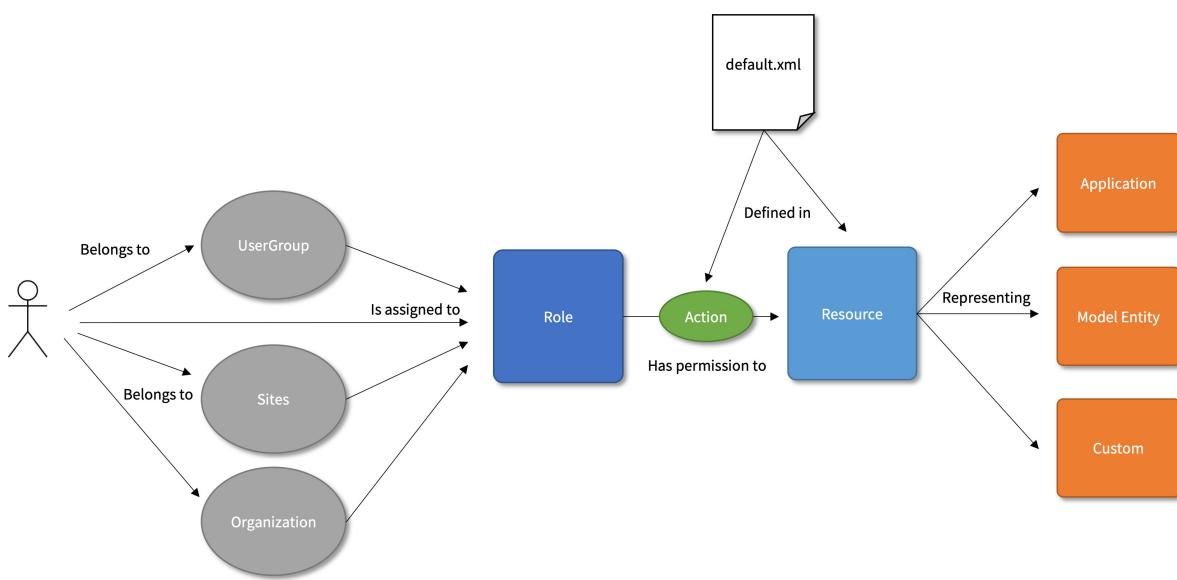
Roles defined with the *Site* or *Organization* scope apply only within the respective site or organization. Examples for Site or Organization roles are the *Site Administrator* or *Organization Administrator* role, which give their members administrative privileges for a certain site or organization.

**Permissions are always granted to roles**, not to users directly.

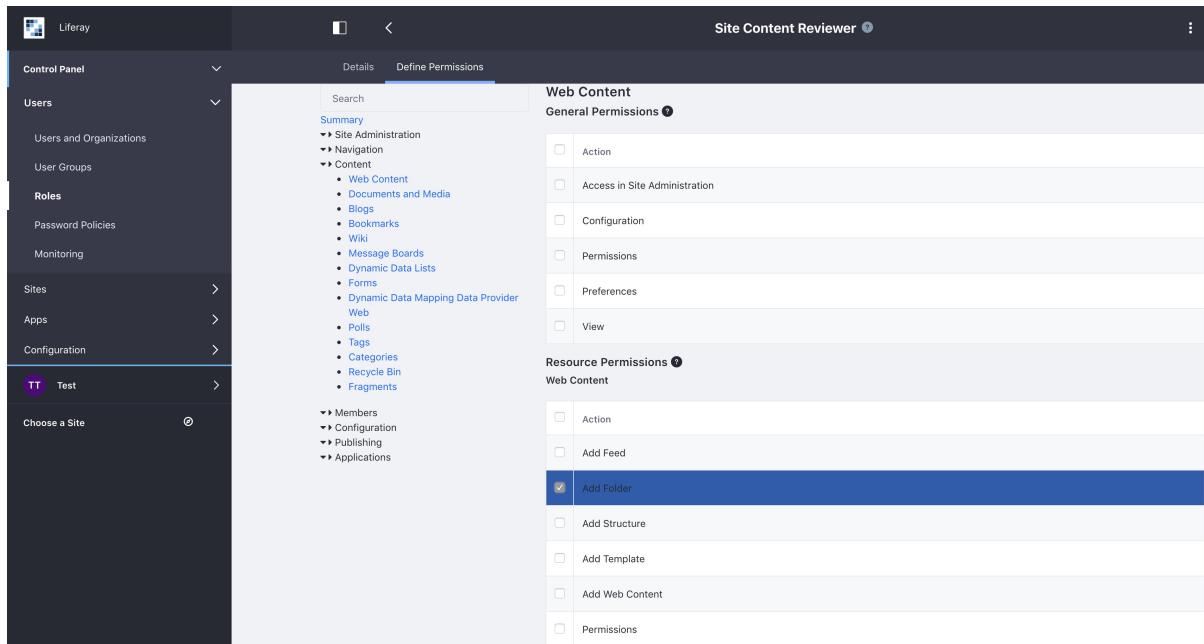
## Wrapping it Up

Liferay's access control system can be summarized as follows:

- **Resources** represent either an application or a model entity.
- **Permissions** are **actions** that can be performed on a resource.
- **Roles** are collections of *permissions* that define a particular function within Liferay.
- Roles are assigned to users either directly or through a site, an organization, or a user group.
- Effective permissions are inherited from all the user's roles, so a Liferay user has all the permissions of all roles they belong to.



Permissions for roles can be granted on an individual resource level. You can manage roles in *Control Panel → Roles*.



## Permissions in the Database

There are two database tables storing the permissions information:

- ResourceAction
- ResourcePermission

Example of Blogs-related actions in the ResourceAction table:

```
mysql> select * from resourceaction where name like '%blogs%' limit 5;
+-----+-----+-----+-----+-----+
| mvccVersion | resourceActionId | name | actionId | bitwiseValue |
+-----+-----+-----+-----+-----+
| 0 | 4 | com.liferay.blogs | ADD_ENTRY | 2 |
| 0 | 5 | com.liferay.blogs | PERMISSIONS | 4 |
| 0 | 6 | com.liferay.blogs | SUBSCRIBE | 8 |
| 0 | 35 | com.liferay.blogs.kernel.model.BlogsEntry | UPDATE_DISCUSSION | 2 |
| 0 | 36 | com.liferay.blogs.kernel.model.BlogsEntry | DELETE | 4 |
+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)
```

Example of Blogs-related permissions in the ResourcePermissions table:

```
mysql> select * from resourcepermission where name like '%blogs%' order by resourcepermissionId desc limit 5;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| mvccVersion | resourcePermsId | companyId | name | scope | primKey | primKeyId | roleId | ownerId | actionIds | viewType | actionId |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 8 | 4406 | 20115 | com.liferay.blogs | 4 | 66680 | 66680 | 20136 | 0 | 8 | 0 |
| 8 | 4405 | 20115 | com.liferay.blogs | 4 | 66680 | 66680 | 20123 | 0 | 14 | 0 |
| 8 | 4320 | 20115 | com.liferay.blogs.kernel.model.BlogsEntry | 4 | 63341 | 63341 | 20122 | 0 | 65 | 1 |
| 8 | 4319 | 20115 | com.liferay.blogs.kernel.model.BlogsEntry | 4 | 63341 | 63341 | 20130 | 0 | 65 | 1 |
| 8 | 4318 | 20115 | com.liferay.blogs.kernel.model.BlogsEntry | 4 | 63341 | 63341 | 20123 | 20155 | 127 | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)
```

## Implementing Permissioning

To implement permissioning, you need to define permissions, manage the resource lifecycle, and implement permission checking. Where and how you define permissions depends on the application structure.

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

Generally, the steps for implementing permissioning are:

1. Define the path to the file that defines the resources and permissions (`default.xml`). This is usually done in the `portlet.properties` file.
2. Define resources and permissions. This is usually done in a file called `default.xml`.
3. Manage the permissions of a resource lifecycle. On the service layer, this is usually done in the CRUD methods of the service implementation class.
4. Create permission registrar classes.
5. Implement the permission checker class if necessary.
6. Implement permission checking wherever necessary.

Let's walk through an example:

## Step 1 - Define the Path to the Permission Definition File

Create a file called `portlet.properties` in the `src/main/resources` folder of your module project and implement as follows:

```

Input a list of comma delimited resource action configurations
that will be
read from the class path.

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

## Step 2 - Create the Permission Definition File

Now you have to implement the actual file that defines module resources and the available permissions. Following the example, this file would be

```
src/main/resources/resource-actions/default.xml .
```

As an example, below are permission definitions in a Liferay Blogs application. The first one defines the portlet permissions for two portlets in the *blogs-web* module. Notice that there are only *portlet-resource* definitions.

The second one is the permission definition file for model resources and is located in the *blogs-service* module. Take note of three things in particular:

1. *model-resource* tags instead of *portlet-resource* tags
2. Resources that have the *model-name* `com.liferay.blogs` are *top-level actions*, which are applied to generic model actions.
3. Resources have the full class path `com.liferay.blogs.model.BlogsEntry` as the *model-name* and are called *resource actions* and are applied to existing entities.

### **blogs-web default.xml**

```
<?xml version="1.0"?>
<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource Action Mapping 7.0.0//EN" "http://www.liferay.com/dtd/liferay-resource-action-mapping_7_0_0.dtd">

<resource-action-mapping>
 <portlet-resource>
 <portlet-name>com_liferay_blogs_web_portlet_BlogsAdminPortlet</portlet-name>
 <permissions>
 <supports>
 <action-key>ACCESS_IN_CONTROL_PANEL</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>ACCESS_IN_CONTROL_PANEL</action-key>
 <action-key>CONFIGURATION</action-key>
 </guest-unsupported>
 </permissions>
 </portlet-resource>
</resource-action-mapping>
```

```

 </permissions>
 </portlet-resource>
 <portlet-resource>
 <portlet-name>com_liferay_blogs_web_portlet_BlogsPortlet</portlet-name>
 <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>CONFIGURATION</action-key>
 </guest-unsupported>
 </permissions>
 </portlet-resource>
</resource-action-mapping>

```



### blogs-service default.xml

```

<?xml version="1.0"?>
<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource Action Mapping 7.0.0//EN" "http://www.liferay.com/dtd/liferay-resource-action-mapping_7_0_0.dtd">

<resource-action-mapping>
 <model-resource>
 <model-name>com.liferay.blogs</model-name>
 <portlet-ref>

```

```
<portlet-name>com_liferay_blogs_web_portlet_BlogsAdmi
nPortlet</portlet-name>
 <portlet-name>com_liferay_blogs_web_portlet_BlogsPort
let</portlet-name>
 </portlet-ref>
 <root>true</root>
 <weight>1</weight>
 <permissions>
 <supports>
 <action-key>ADD_ENTRY</action-key>
 <action-key>PERMISSIONS</action-key>
 <action-key>SUBSCRIBE</action-key>
 </supports>
 <site-member-defaults>
 <action-key>SUBSCRIBE</action-key>
 </site-member-defaults>
 <guest-defaults />
 <guest-unsupported>
 <action-key>ADD_ENTRY</action-key>
 <action-key>PERMISSIONS</action-key>
 <action-key>SUBSCRIBE</action-key>
 </guest-unsupported>
 </permissions>
 </model-resource>
 <model-resource>
 <model-name>com.liferay.blogs.model.BlogsEntry</model-name
>
 <portlet-ref>
 <portlet-name>com_liferay_blogs_web_portlet_BlogsAdmi
nPortlet</portlet-name>
 <portlet-name>com_liferay_blogs_web_portlet_BlogsPort
let</portlet-name>
 </portlet-ref>
 <weight>2</weight>
 <permissions>
 <supports>
 <action-key>ADD_DISCUSSION</action-key>
 <action-key>DELETE</action-key>
 <action-key>DELETE_DISCUSSION</action-key>
 <action-key>PERMISSIONS</action-key>
 </supports>
```

```

<action-key>UPDATE</action-key>
<action-key>UPDATE_DISCUSSION</action-key>
<action-key>VIEW</action-key>
</supports>
<site-member-defaults>
<action-key>ADD_DISCUSSION</action-key>
<action-key>VIEW</action-key>
</site-member-defaults>
<guest-defaults>
<action-key>ADD_DISCUSSION</action-key>
<action-key>VIEW</action-key>
</guest-defaults>
<guest-unsupported>
<action-key>DELETE</action-key>
<action-key>DELETE_DISCUSSION</action-key>
<action-key>PERMISSIONS</action-key>
<action-key>UPDATE</action-key>
<action-key>UPDATE_DISCUSSION</action-key>
</guest-unsupported>
</permissions>
</model-resource>
</resource-action-mapping>

```



Sources:

- <https://github.com/liferay/liferay-portal/blob/7.2.x/modules/apps/blogs/blogs-web/src/main/resources/resource-actions/default.xml>
- <https://github.com/liferay/liferay-portal/blob/7.2.x/modules/apps/blogs/blogs-service/src/main/resources/resource-actions/default.xml>

## Step 3 - Manage the Permission Resources Lifecycle

Permission resources are permission container objects bound to the model entities. We need to take care of adding, updating, and deleting permission resources for our entities.

In a Service Builder project, adding is usually done in the method adding a new entity and deletion respectively in the entity deletion method. Portal services for managing permission resources are `com.liferay.portal.kernel.service.ResourceService` and

```
com.liferay.portal.kernel.service.ResourceLocalService .
```

Below is an excerpt of resource handling methods in a Blogs application's local service class: [com.liferay.blogs.service.impl.BlogsEntryLocalServiceImpl.java](#):

## Adding resources

```
@Override
public void addEntryResources(
 BlogsEntry entry, boolean addGroupPermissions,
 boolean addGuestPermissions)
throws PortalException {

 resourceLocalService.addResources(
 entry.getCompanyId(), entry.getGroupId(), entry.getUserId()
,
 BlogsEntry.class.getName(), entry.getEntryId(), false,
 addGroupPermissions, addGuestPermissions);
}

@Override
public void addEntryResources(
 BlogsEntry entry, ModelPermissions modelPermissions)
throws PortalException {

 resourceLocalService.addModelResources(
 entry.getCompanyId(), entry.getGroupId(), entry.getUserId()
,
 BlogsEntry.class.getName(), entry.getEntryId(), modelPerm
issions);
}
```

## Updating resources

```
@Override
public void updateEntryResources(
 BlogsEntry entry, ModelPermissions modelPermissions)
throws PortalException {
```

```

 resourceLocalService.updateResources(
 entry.getCompanyId(), entry.getGroupId(),
 BlogsEntry.class.getName(), entry.getEntryId(), modelPermissions);
 }

@Override
public void updateEntryResources(
 BlogsEntry entry, String[] groupPermissions,
 String[] guestPermissions)
throws PortalException {

 resourceLocalService.updateResources(
 entry.getCompanyId(), entry.getGroupId(),
 BlogsEntry.class.getName(), entry.getEntryId(), groupPermissions,
 guestPermissions);
}

```

## Resource Deletion

```

@Indexable(type = IndexableType.DELETE)
@Override
@SystemEvent(type = SystemEventConstants.TYPE_DELETE)
public BlogsEntry deleteEntry(BlogsEntry entry) throws PortalException {

 // Order is important. See LPS-81826.

 // Ratings

 ratingsStatsLocalService.deleteStats(
 BlogsEntry.class.getName(), entry.getEntryId());

 // Entry

 blogsEntryPersistence.remove(entry);

 // Resources
}

```

```
resourceLocalService.deleteResource(
 entry.getCompanyId(), BlogsEntry.class.getName(),
 ResourceConstants.SCOPe_INDIVIDUAL, entry.getEntryId());

// Image

imageLocalService.deleteImage(entry.getSmallImageId());

// Subscriptions

subscriptionLocalService.deleteSubscriptions(
 entry.getCompanyId(), BlogsEntry.class.getName(),
 entry.getEntryId());

// Statistics

blogsStatsUserLocalService.updateStatsUser(
 entry.getGroupId(), entry.getUserId(), entry.getDisplayDa
te());

// Asset

assetEntryLocalService.deleteEntry(
 BlogsEntry.class.getName(), entry.getEntryId());

// Attachments

long coverImageFileEntryId = entry.getCoverImageFileEntryId()
;

if (coverImageFileEntryId != 0) {
 PortletFileRepositoryUtil.deletePortletFileEntry(
 coverImageFileEntryId);
}

long smallImageFileEntryId = entry.getSmallImageFileEntryId()
;

if (smallImageFileEntryId != 0) {
```

```

 PortletFileRepositoryUtil.deletePortletFileEntry(
 smallImageFileEntryId);
 }

 // Comment

 deleteDiscussion(entry);

 // Expando

 expandoRowLocalService.deleteRows(entry.getEntryId());

 // Friendly URL

 friendlyURLEntryLocalService.deleteFriendlyURLEntry(
 entry.getGroupId(), BlogsEntry.class, entry.getEntryId())
;

 // Trash

 trashEntryLocalService.deleteEntry(
 BlogsEntry.class.getName(), entry.getEntryId());

 // Workflow

 workflowInstanceLinkLocalService.deleteWorkflowInstanceLinks(
 entry.getCompanyId(), entry.getGroupId(),
 BlogsEntry.class.getName(), entry.getEntryId());

 return entry;
}

```

## Step 4 - Create Permission Registrar Classes

Next, we'll have to create permissions registrar classes. These follow what you did in `default.xml` : you need one for your portlet permissions and one for each of your entities. By convention, registrar classes are placed in the sub-package `internal.security.permission.resource` .

**BlogsEntryModelResourcePermissionDefinition.java**

```
/*
 * 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 Li
 * cense for more
 * details.
 */

package com.liferay.blogs.internal.security.permission.resource.d
efinition;

import com.liferay.blogs.constants.BlogsConstants;
import com.liferay.blogs.constants.BlogsPortletKeys;
import com.liferay.blogs.model.BlogsEntry;
import com.liferay.blogs.service.BlogsEntryLocalService;
import com.liferay.exportimport.kernel.staging.permission.Staging
Permission;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.resource.Mod
elResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.Mod
elResourcePermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.Por
tletResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.Sta
gedModelPermissionLogic;
```

```
import com.liferay.portal.kernel.security.permission.resource.Wor
kflowedModelPermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.def
inition.ModelResourcePermissionDefinition;
import com.liferay.portal.kernel.service.GroupLocalService;
import com.liferay.portal.kernel.workflow.permission.WorkflowPerm
ission;

import java.util.function.Consumer;

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

/**
 * @author Preston Crary
 */
@Component(immediate = true, service = ModelResourcePermissionDef
inition.class)
public class BlogsEntryModelResourcePermissionDefinition
 implements ModelResourcePermissionDefinition<BlogsEntry> {

 @Override
 public BlogsEntry getModel(long entryId) throws PortalException {
 return _blogsEntryLocalService.getBlogsEntry(entryId);
 }

 @Override
 public Class<BlogsEntry> getModelClass() {
 return BlogsEntry.class;
 }

 @Override
 public PortletResourcePermission getPortletResourcePermission
() {
 return _portletResourcePermission;
 }

 @Override
 public long getPrimaryKey(BlogsEntry blogsEntry) {
```

```

 return blogsEntry.getEntryId();
 }

 @Override
 public void registerModelResourcePermissionLogics(
 ModelResourcePermission<BlogsEntry> modelResourcePermission,
 Consumer<ModelResourcePermissionLogic<BlogsEntry>>
 modelResourcePermissionLogicConsumer) {

 modelResourcePermissionLogicConsumer.accept(
 new StagedModelPermissionLogic<>(
 _stagingPermission, BlogsPortletKeys.BLOGS,
 BlogsEntry::getEntryId));
 modelResourcePermissionLogicConsumer.accept(
 new WorkflowedModelPermissionLogic<>(
 _workflowPermission, modelResourcePermission,
 _groupLocalService, BlogsEntry::getEntryId));
 }

 @Reference
 private BlogsEntryLocalService _blogsEntryLocalService;

 @Reference
 private GroupLocalService _groupLocalService;

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

 @Reference
 private StagingPermission _stagingPermission;

 @Reference
 private WorkflowPermission _workflowPermission;

}

```

**BlogsPortletResourcePermissionDefinition.java**

```
/***
 * 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 Li
 * cense for more
 * details.
 */
package com.liferay.blogs.internal.security.permission.resource.d
efinition;

import com.liferay.blogs.constants.BlogsConstants;
import com.liferay.blogs.constants.BlogsPortletKeys;
import com.liferay.exportimport.kernel.staging.permission.Staging
Permission;
import com.liferay.portal.kernel.security.permission.resource.Po
r
tletResourcePermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.Sta
gedPortletPermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.def
inition.PortletResourcePermissionDefinition;

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

/**
 * @author Preston Crary
 */
```

```

@Component(
 immediate = true, service = PortletResourcePermissionDefinition.class
)
public class BlogsPortletResourcePermissionDefinition
 implements PortletResourcePermissionDefinition {

 @Override
 public PortletResourcePermissionLogic[]
 getPortletResourcePermissionLogics() {

 return new PortletResourcePermissionLogic[] {
 new StagedPortletPermissionLogic(
 _stagingPermission, BlogsPortletKeys.BLOGS_ADMIN)
 };
 }

 @Override
 public String getResourceName() {
 return BlogsConstants.RESOURCE_NAME;
 }

 @Reference
 private StagingPermission _stagingPermission;

}

```

Sources:

- BlogsPortletResourcePermissionDefinition: <https://github.com/liferay/liferay-portal/blob/7.2.x/modules/apps/blogs/blogs-service/src/main/java/com/liferay/blogs/internal/security/permission/resource/definition/BlogsPortletResourcePermissionDefinition.java>
- BlogsEntryModelResourcePermissionDefinition: <https://github.com/liferay/liferay-portal/blob/7.2.x/modules/apps/blogs/blogs-service/src/main/java/com/liferay/blogs/internal/security/permission/resource/definition/BlogsEntryModelResourcePermissionDefinition.java>

## Step 5 - Implement Permission Checkers

Next, we'll need a permission checker class for portlets and all the entities. In the Liferay Blogs application, both are placed in the web module:

### BlogsPermission.java

```
/*
 * 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 Li
 * cense for more
 * details.
 */

package com.liferay.blogs.web.internal.security.resour
ce;

import com.liferay.blogs.constants.BlogsConstants;
import com.liferay.portal.kernel.security.permission.PermissionCh
ecker;
import com.liferay.portal.kernel.security.permission.resource.Por
tletResourcePermission;

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

```

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

/**
 * @author Preston Crary
 */
@Component(immediate = true)
public class BlogsPermission {

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

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

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

 _portletResourcePermission = portletResourcePermission;
 }

 private static PortletResourcePermission _portletResourcePerm
ission;

}

```

**BlogsEntryPermission.java**

```

* 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

```

```
d 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
LITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public Li
cense for more
 * details.
*/
package com.liferay.blogs.web.internal.security.permission.resour
ce;

import com.liferay.blogs.model.BlogsEntry;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.PermissionCh
ecker;
import com.liferay.portal.kernel.security.permission.resource.Mod
elResourcePermission;

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

/**
 * @author Preston Crary
 */
@Component(immediate = true, service = BlogsEntryPermission.class
)
public class BlogsEntryPermission {

 public static boolean contains(
 PermissionChecker permissionChecker, BlogsEntry entry
 ,
 String actionId)
 throws PortalException {
```

```

 return _blogsEntryModelResourcePermission.contains(
 permissionChecker, entry, actionId);
 }

 public static boolean contains(
 PermissionChecker permissionChecker, long entryId, St
ring actionId)
 throws PortalException {

 return _blogsEntryModelResourcePermission.contains(
 permissionChecker, entryId, actionId);
 }

 @Reference(
 target = "(model.class.name=com.liferay.blogs.model.Blogs
Entry)",
 unbind = "-"
)
 protected void setEntryModelPermission(
 ModelResourcePermission<BlogsEntry> modelResourcePermissi
on) {

 _blogsEntryModelResourcePermission = modelResourcePermiss
ion;
 }

 private static ModelResourcePermission<BlogsEntry>
 _blogsEntryModelResourcePermission;
}

```

Sources:

- [BlogsEntryPermission](#)
- [BlogsPermission](#)

## Step 6 - Implement Permission Checking

The final step is to make use of the permissions.

In the user interface, you should check permissions to control viewing, adding, and deleting entities. You also have to implement permission checking on the service layer, as services can be accessed outside your application user interface. The permission checking should be done on your remote service implementation class. This is why user-level access to services should, therefore, always be done through remote service classes. Local service classes **should not** implement permission checking.

The examples below from a Blogs applications `BlogsEntryServiceImpl` and from a `JSP` file class demonstrates permission checking:

### **BlogsEntryServiceImpl.java**

```
@Override
public BlogsEntry addEntry(
 String title, String subtitle, String description, String
content,
 int displayDateMonth, int displayDateDay, int displayDate
Year,
 int displayDateHour, int displayDateMinute, boolean allow
Pingbacks,
 boolean allowTrackbacks, String[] trackbacks,
 String coverImageCaption, ImageSelector coverImageImageSe
lector,
 ImageSelector smallImageImageSelector,
 ServiceContext serviceContext)
throws PortalException {

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

 return blogsEntryLocalService.addEntry(
 getUserId(), title, subtitle, description, content,
 displayDateMonth, displayDateDay, displayDateYear, displa
yDateHour,
 displayDateMinute, allowPingbacks, allowTrackbacks, track
backs,
 coverImageCaption, coverImageImageSelector, smallImageIma
geSelector,
 serviceContext);
```

```
}
```

### entry\_action.jsp

```
<c:if test="<%= BlogsEntryPermission.contains(permissionChecker, entry, ActionKeys.PERMISSIONS) %>">
 <liferay-security:permissionsURL
 modelResource="<%= BlogsEntry.class.getName() %>"
 modelResourceDescription="<%= BlogsEntryUtil.getDisplayTitle(resourceBundle, entry) %>"
 resourceGroupId="<%= String.valueOf(entry.getGroupId()) %>"
 resourcePrimKey="<%= String.valueOf(entry.getEntryId()) %>"
 var="permissionsEntryURL"
 windowState="<%= LiferayWindowState.POP_UP.toString() %>">
 </liferay-security:permissionsURL>
 <liferay-ui:icon
 label="<%= true %>"
 message="permissions"
 method="get"
 url="<%= permissionsEntryURL %>"
 useDialog="<%= true %>">
 </liferay-ui:icon>
</c:if>
```

## Knowledge Check

- Permissions are \_\_\_\_\_ that can be performed on a given \_\_\_\_\_.
- Roles are collections of \_\_\_\_\_.
- To implement permissioning in your application:
  - Define \_\_\_\_\_ to the permission definition file.
  - Define \_\_\_\_\_ and \_\_\_\_\_.
  - Manage \_\_\_\_\_ lifecycle.
  - Implement permission \_\_\_\_\_ classes.
  - Implement permission \_\_\_\_\_ classes, if necessary.
  - Implement permission \_\_\_\_\_ where ever necessary.

## Exercises

# 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 have to **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.

### Create the Resource Constants Class

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

#### 1. Create a class

`com.liferay.training.gradebook.constants.GradebookConstants` and implement as follows:

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

```
public class GradebookConstants {
```

```

    public static final String RESOURCE_NAME = "com.liferay.training.gradebook.model";
}
```

```

1. **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:\n
 com.liferay.training.gradebook.constants,\n
 com.liferay.training.gradebook.exception,\n
 com.liferay.training.gradebook.model,\n
 com.liferay.training.gradebook.service,\n
 com.liferay.training.gradebook.service.persistence,\n
 com.liferay.training.gradebook.validator\n
-check: EXPORTS\n
-includeresource: META-INF/service.xml=../gradebook-service/service.xml

```

## Define the Permissions

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

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 7.2.0//EN" "http://www.liferay.com/dtd/liferay-resource-action-mapping_7_2_0.dtd">
<resource-action-mapping>
 <model-resource>
 <model-name>com.liferay.training.gradebook.model</mod

```

```
el-name>
 <portlet-ref>
 <portlet-name>com_liferay_training_gradebook_web_
portlet_GradebookPortlet</portlet-name>
 </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.Assi
gnment</model-name>
 <portlet-ref>
 <portlet-name>com_liferay_training_gradebook_web_
portlet_GradebookPortlet</portlet-name>
 </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
```

## Implement Permission Resource Management

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:

1. **Open** the class

```
com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl .
```

2. Replace the `addAssignment()` method with the following. Notice the highlighted rows:

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

 // Validate assignment parameters.

 _assignmentValidator.validate(titleMap, descriptionMap, d
 ueDate);

 // 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 assignment. 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.setDescriptionMap(descriptionMap);
 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.cla
ss.getName(),
 assignment.getAssignmentId(), portletActions, addGrou
pPermissions,
 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);
}

```

```
}
```

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

4. **Resolve** missing imports.

## Create the Permission Registrar Classes

Create classes for registering the model and top-level resource permissions:

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.resource.definition;

import com.liferay.exportimport.kernel.staging.permission.StagingPermission;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.PortletResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.StagedModelPermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.WorkflowedModelPermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.definition.ModelResourcePermissionDefinition;
import com.liferay.portal.kernel.service.GroupLocalService;
import com.liferay.portal.kernel.workflow.permission.WorkflowPermission;
import com.liferay.training.gradebook.constants.GradebookCons
```

```
tants;
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>> modelResourcePermissionLogicConsumer) {

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

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

 // modelResourcePermissionLogicConsumer.accept(
 // new WorkflowedModelPermissionLogic<>(
 // _workflowPermission, modelResourcePermission,
 // // _groupLocalService, Assignment::getAssignmentId));
}

@Reference
private AssignmentLocalService _assignmentLocalService;

@Reference
private GroupLocalService _groupLocalService;

@Reference(target = "(resource.name=" + GradebookConstants.RESOURCE_NAME + ")")

```

```

 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.definition.AssignmentPortletResourcePermissionDefinition
```

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.PortletResourcePermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.StagedPortletPermissionLogic;
import com.liferay.portal.kernel.security.permission.resource.definition.PortletResourcePermissionDefinition;
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_GradebookPortlet")
 };
 }

 @Override
 public String getResourceName() {

 return GradebookConstants.RESOURCE_NAME;
 }

 @Reference
 private StagingPermission _stagingPermission;

}

```

For more technical information about registrar classes see [Liferay Developer Network](#)

## Implement Permission Checking in the Remote Service

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.mo
del.Assignment)"
)
private volatile ModelResourcePermission<Assignment>
_assignmentModelResourcePermission;

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

```

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.

Now 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 Li
 * cense 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.ModelResourcePermissionFactory;
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, rerun ServiceBuilder to copy their definitions into the <code>com.liferay.training.gradebook.service.AssignmentService</code> interface.
 *
 * <p>
 * This is a remote service. Methods of this service are expected to have security checks based on the propagated JAAS credentials

```

```
because this service can be accessed remotely.
* </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 AssignmentServiceBaseI
mpl {

 /*
 * NOTE FOR DEVELOPERS:
 *
 * Never reference this class directly. Always use <code>com.
 * liferay.training.gradebook.service.AssignmentServiceUtil</code> t
 * o access the assignment remote service.
 */
 public Assignment addAssignment(
 long groupId, Map<Locale, String> titleMap, Map<Locale, S
tring> descriptionMap,
 Date dueDate, ServiceContext serviceContext)
 throws PortalException {

 // Check permissions.

 _portletResourcePermission.check(
 getPermissionChecker(), serviceContext.getScopeGroupI
d(),
 ActionKeys.ADD_ENTRY);

 return assignmentLocalService.addAssignment(
 groupId, titleMap, descriptionMap, dueDate, serviceCo
ntext);
```

```
}

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, Map<Locale, String> descriptionMap,
 Date dueDate, ServiceContext serviceContext)
 throws PortalException {

 // Check permissions.

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

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

 @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 _portletResourcePe
rmission;
}

```



## 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:



GRADEBOOK

**① Error:** Service request failed with message: com.liferay.portal.kernel.security.auth.PrincipalException\$MustHavePermission: User 20103 must have ADD\_ENTRY permission for com.liferay.training.gradebook.model 20126

### Add New Assignment

Title \*

Am I authorized?



Description \*

Description

Due Date \*

04/05/2019

01:40 PM

**Save****Cancel**

## Exercises

# 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 Res
ource Action apping 7.2.0//EN" "http://www.liferay.com/dtd/li
ferry-resource-action-mapping_7_2_0.dtd">
<resource-action-mapping>
 <portlet-resource>
 <portlet-name>com_liferay_training_gradebook_web_port
let_GradebookPortlet</portlet-name>
 <permissions>
 <supports>
 <action-key>ADD_PORTLET_DISPLAY_TEMPLATE</act
ion-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</act
ion-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 the Top-Level Resource Permission Checker Class

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.

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 _portletResource
Permission;

}

```

## Implement the Model Resource Permission Checker Class

Implement a class for checking existing entity permissions.

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 a
}

```

```

ssignment,
 String actionId)
throws PortalException {

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

public static boolean contains(
 PermissionChecker permissionChecker, long assignm
entId, String actionId)
throws PortalException {

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

@Reference(
 target = "(model.class.name=com.liferay.training.grad
ebook.model.Assignment)",
 unbind = "-"
)
protected void setEntryModelPermission(
 ModelResourcePermission<Assignment> modelResourcePerm
ission) {

 _assignmentModelResourcePermission = modelResourcePer
mission;
}

private static ModelResourcePermission<Assignment>
_assignmentModelResourcePermission;
}

```

## Implement Permission Checking in the JSP Files

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.

1. **Open** the class

```
com.liferay.training.gradebook.web.portlet.action.ViewAssignment
sMVCRenderCommand
```

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 will 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.Ass
signmentsManagementToolbarDisplayContext;
import com.liferay.training.gradebook.web.internal.security.p
ermission.resource.AssignmentPermission;

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.Gradeboo
k,
 "mvc.command.name=/",
 "mvc.command.name=" + MVCCCommandNames.VIEW_ASSIGNMENT
S
 },
 service = MVCRenderCommand.class
)
public class ViewAssignmentsMVCRenderCommand implements MVCRe
nderCommand {

 @Override
 public String render(
 RenderRequest renderRequest, RenderResponse renderRes
ponse)
 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 assignment 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, sta
rt, end,
 comparator);

 // Set request attributes.
```

```
 renderRequest.setAttribute("assignments", assignments)
;
 renderRequest.setAttribute(
 "assignmentCount", _assignmentService.getAssignmentsCountByKeywords(
 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 assignmentsManagementToolbarDisplayContext =
 new AssignmentsManagementToolbarDisplayContext(
 liferayPortletRequest, liferayPortletResponse
,
 _portal.getHttpServletRequest(renderRequest))
;

 renderRequest.setAttribute(
 "assignmentsManagementToolbarDisplayContext",
 assignmentsManagementToolbarDisplayContext);
}
```

```

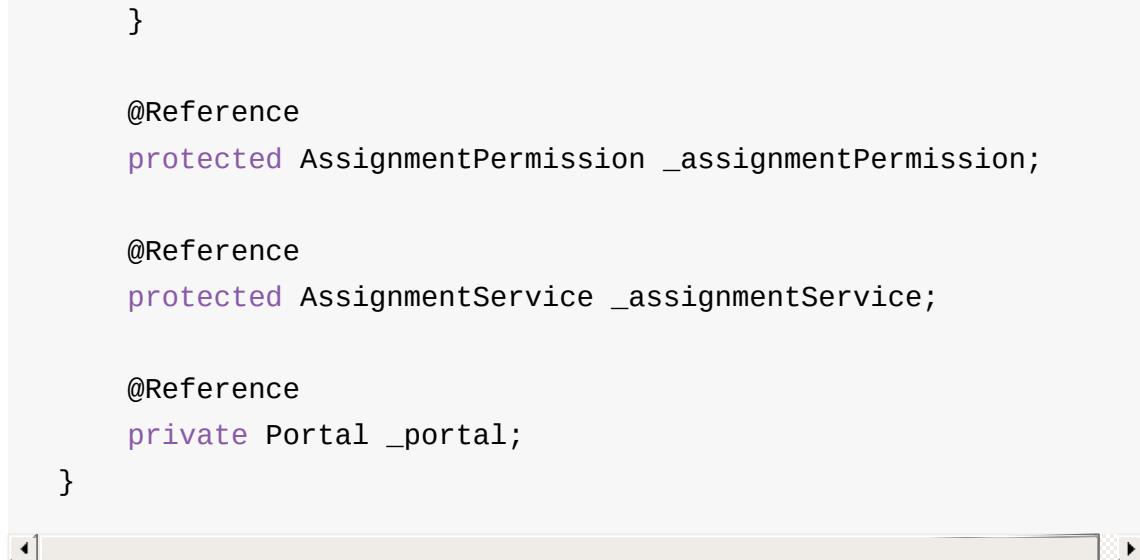
 }

 @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:

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. Notice the highlighted rows:

```

<%@ 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(permissionChe
cker, assignment.assignmentId, 'VIEW')}">
 <portlet:renderURL var="viewAssignmentURL">
 <portlet:param name="mvcRenderCommandName"
 value="<%="MVCCommandNames.VIEW_ASSIGNMENT %>">
 />
 <portlet:param name="redirect" value="\${currentUR
L}" />
 <portlet:param name="assignmentId" value="\${assi
gnment.assignmentId}" />
</portlet:renderURL>

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

<%-- Edit action. --%>

<c:if test="\${assignmentPermission.contains(permissionChe
cker, assignment.assignmentId, 'UPDATE')}">
 <portlet:renderURL var="editAssignmentURL">
 <portlet:param name="mvcRenderCommandName"
 value="<%="MVCCommandNames.EDIT_ASSIGNMENT %>">
 />
 <portlet:param name="redirect" value="\${currentUR
L}" />
 <portlet:param name="assignmentId" value="\${assi
gnment.assignmentId}" />
</portlet:renderURL>

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

<%-- Permissions action. --%>

<c:if test="\${assignmentPermission.contains(permissionChe
cker, assignment.assignmentId, 'PERMISSIONS')}">

```

```

 <liferay-security:permissionsURL
 modelResource="com.liferay.training.gradebook.mod
el.Assignment"
 modelResourceDescription="${assignment.getTitle(1
ocale)}"
 resourcePrimKey="${assignment.assignmentId}"
 var="permissionsURL"
 />

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

 <%-- Delete action. --%>

 <c:if test="${assignmentPermission.contains(permissionChe
cker, assignment.assignmentId, 'DELETE')}">

 <portlet:actionURL name="<%="MVCCommandNames.DELETE_AS
SIGNMENT %>" var="deleteAssignmentURL">
 <portlet:param name="redirect" value="${currentUR
L}" />
 <portlet:param name="assignmentId" value="${assi
gnment.assignmentId}" />
 </portlet:actionURL>

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

```

## Implement Permission Checking in the Management Toolbar

The last thing 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:

1. Open the class

```
com.liferay.training.gradebook.web.display.context.AssignmentsMa
```

nagementToolbarDisplayContext.java .

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

```

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.createRenderUR
L(),
 "mvcRenderCommandName", MVCCommandNam
es.EDIT_ASSIGNMENT,
 "redirect", currentURLObj.toString())
 ;
 dropdownItem.setLabel(
 LanguageUtil.get(request, "add-assign
ment"));
 });
 }
 };
}

```

3. **Resolve** missing imports.

## Test the Application

1. **Sign out** and test whether you can add, edit, or delete Assignments.
2. **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.

# Integrate with the Asset Framework

The Asset Framework is a Liferay platform framework that makes it possible to publish and manage any kind of content in a unified way and through a standard API. The framework provides ways of associating and linking content with, for example, other portal assets, tags, and categories, and makes it possible to integrate with portal search, workflows and staging.

The central concepts in integrating with the Asset Framework are an Asset, an Asset Renderer, and an Asset Renderer Factory.

## Assets

An Asset is an abstract, generic representation of any model entity wrapped in an [AssetEntry](#) class guaranteeing a certain set of metadata (fields) for the consuming applications and APIs.

## Asset Renderers

The Asset Renderer Class is responsible for rendering the URLs for viewing and editing an asset, checking view permissions, and providing access to the wrapped entity. The Asset Renderer implements the [AssetRenderer](#) interface.

## Asset Renderer Factories

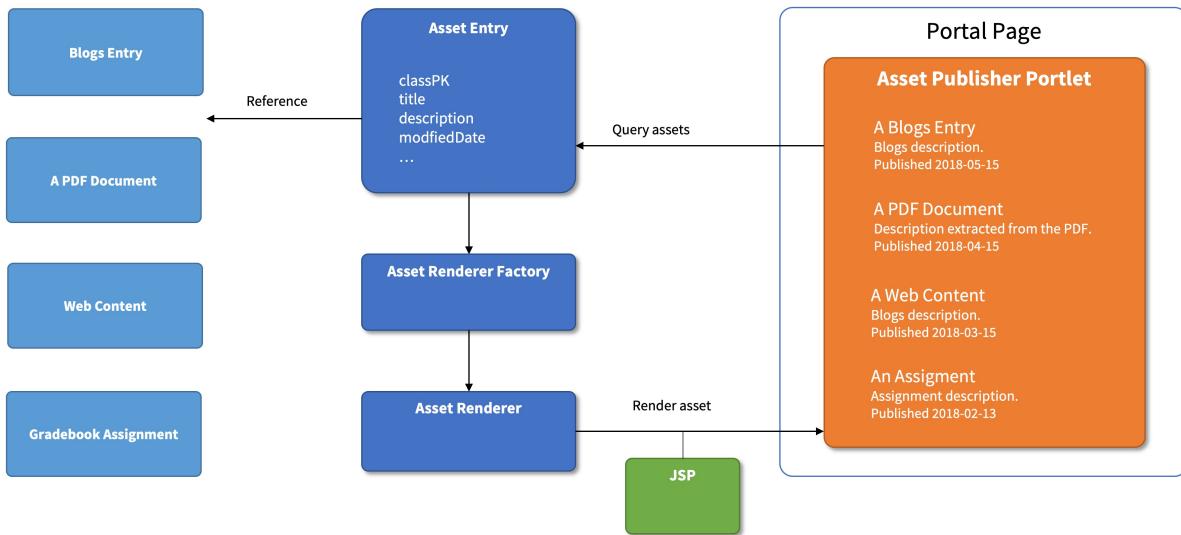
An Asset Renderer Factory is an OSGi component class that makes an Asset Renderer available to the calling application or API. The factory pattern provides a possibility to have multiple renderers for a single asset type.

## Asset Framework Diagram

The diagram below illustrates and summarizes the components of the Asset Framework.

On a portal page, there is an **Asset Publisher** portlet querying the newest assets. The Asset Framework gets a list of assets that contain the required set of metadata and references to the actual content items. When rendering the assets, the Asset Publisher portlet first finds the **Asset Renderer Factory** service for the model type and then asks for an **Asset**

**Renderer** from the factory. The Asset Renderer uses the data from the actual content item, wrapped by an asset, to render the item on the type specific **JSP files** provided. Every Asset Publisher view, like abstracts, full content, and table has a dedicated JSP file:



## An Asset in the Database

Below is an example of database table structures for BlogsEntry and AssetEntry. You can see that the AssetEntry contains a subset of the data of BlogsEntry:

### BlogsEntry

```

mysql> describe blogsentry;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | E
xtra |
+-----+-----+-----+-----+-----+
| uuid_ | varchar(75) | YES | MUL | NULL |
| entryId | bigint(20) | NO | PRI | NULL |
| groupId | bigint(20) | YES | MUL | NULL |
| companyId | bigint(20) | YES | MUL | NULL |
| ...

```

userId	bigint(20)	YES		NULL	
userName	varchar(75)	YES		NULL	
createDate	datetime(6)	YES		NULL	
modifiedDate	datetime(6)	YES		NULL	
title	varchar(150)	YES		NULL	
subtitle	longtext	YES		NULL	
urlTitle	varchar(150)	YES		NULL	
description	longtext	YES		NULL	
content	longtext	YES		NULL	
displayDate	datetime(6)	YES	MUL	NULL	
allowPingbacks	tinyint(4)	YES		NULL	
allowTrackbacks	tinyint(4)	YES		NULL	
trackbacks	longtext	YES		NULL	
coverImageCaption	longtext	YES		NULL	
coverImageFileEntryId	bigint(20)	YES		NULL	
coverImageURL	longtext	YES		NULL	
smallImage	tinyint(4)	YES		NULL	
smallImageFileEntryId	bigint(20)	YES		NULL	
smallImageId	bigint(20)	YES		NULL	
smallImageURL	longtext	YES		NULL	

```

| lastPublishDate | datetime(6) | YES | | NULL |
|
| status | int(11) | YES | | NULL |
|
| statusByUserId | bigint(20) | YES | | NULL |
|
| statusByUserName | varchar(75) | YES | | NULL |
|
| statusDate | datetime(6) | YES | | NULL |
|
+-----+-----+-----+-----+-----+
-----+
29 rows in set (0,01 sec)

```

## AssetEntry

```

mysql> describe assetentry;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| entryId | bigint(20) | NO | PRI | NULL | |
| groupId | bigint(20) | YES | MUL | NULL | |
| companyId | bigint(20) | YES | MUL | NULL | |
| userId | bigint(20) | YES | | NULL | |
| userName | varchar(75) | YES | | NULL | |
| createDate | datetime(6) | YES | | NULL | |
| modifiedDate | datetime(6) | YES | | NULL | |
| classNameId | bigint(20) | YES | MUL | NULL | |
| classPK | bigint(20) | YES | | NULL | |
| classUuid | varchar(75) | YES | | NULL | |
| classTypeId | bigint(20) | YES | | NULL | |
| listable | tinyint(4) | YES | | NULL | |
| visible | tinyint(4) | YES | MUL | NULL | |
| startDate | datetime(6) | YES | | NULL | |
| endDate | datetime(6) | YES | | NULL | |
| publishDate | datetime(6) | YES | MUL | NULL | |
| expirationDate | datetime(6) | YES | MUL | NULL | |
| mimeType | varchar(75) | YES | | NULL | |
| title | longtext | YES | | NULL | |

```

```

| description | longtext | YES | | NULL | | |
| summary | longtext | YES | | NULL | | |
| url | longtext | YES | | NULL | | |
| layoutUuid | varchar(75) | YES | MUL | NULL | | |
| height | int(11) | YES | | NULL | | |
| width | int(11) | YES | | NULL | | |
| priority | double | YES | | NULL | | |
| viewCount | int(11) | YES | | NULL | | |
+-----+-----+-----+-----+-----+
27 rows in set (0,00 sec)

```

The wrapped model entity is referenced by its ID in the AssetEntry's `classPK` (Class Primary Key) field. In the example below, the `classPK` contains the `entryId` for a BlogsEntry:

### AssetEntry

```

mysql> select entryId, classPK, groupId, companyId, userId, userName, title from assetentry where classPK=63341;
+-----+-----+-----+-----+-----+-----+
| entryId | classPK | groupId | companyId | userId | userName
| title | |
+-----+-----+-----+-----+-----+-----+
| 63342 | 63341 | 47971 | 20115 | 20155 | Liferay Demo
| New Great Blog Entry |
+-----+-----+-----+-----+-----+-----+
1 row in set (0,00 sec)

```

### BlogsEntry

```

mysql> select entryId, groupId, companyId, userId, userName, title from blogsentry where entryId=63341;
+-----+-----+-----+-----+-----+-----+
| entryId | groupId | companyId | userId | userName | title
+-----+-----+-----+-----+-----+-----+

```

```

| |
+-----+-----+-----+
| 63341 | 47971 | 20115 | 20155 | Liferay Demo | New Gre
at Blog Entry |
+-----+-----+-----+
-----+-----+-----+
1 row in set (0,00 sec)

```

Asset entry fields and their descriptions:

- **userId**: the user updating the content
- **groupId**: the scope group of the created content
- **createDate**: the date the entity was created
- **modifiedDate**: the date the entity was last modified
- **className**: identifies the entity's class
- **classPK**: the primary key of the model entity
- **classUuid**: a secondary identifier that's guaranteed to be universally unique
- **classTypeId**: identifies the particular variation of this class (if any, default 0)
- **categoryIds**: the asset category ids for the entity
- **tagNames**: tag names for the entity
- **listable**: specifies whether the entity can be shown in dynamic lists of content (Asset Publisher)
- **visible**: specifies whether the entity is approved
- **startDate**: when the entity should be visible
- **endDate**: when the entity should stop being visible
- **publishDate**: the date the entity will be published (visible)
- **expirationDate**: the date the entity will be archived (not visible)
- **mimetype**: the Mime type, such as ContentTypes.TEXT\_HTML of the content
- **title**: the entity's name
- **description**: a String-based textual description of the entity
- **summary**: a shortened or truncated sample of the entity's content
- **url**: a URL to optionally associate with the entity
- **layoutUuid**: the universally unique ID of the layout of the entry's default display page
- **height**: this can be set to 0
- **width**: this can be set to 0
- **priority**: specifies how the entity is ranked among peer entity instances; the lower

numbers take priority

## The Benefits of Integrating the Asset Framework

To leverage most of Liferay's native features, a custom entity must be integrated into the Asset Framework. Integration allows you to:

- Show custom entities in the Asset Publisher portlet
- Associate tags and categories
- Associate comments and ratings
- Integrate with portal search
- Integrate with portal workflows
- Enable staging on the entities
- Link assets to each other
- Assign social bookmarks like Facebook likes to the entity
- Add custom fields (Liferay Expando API)
- Track the number of times an asset is viewed
- Implement Recycle Bin support

## Steps for Integrating the Asset Framework

Generally, the steps to integrate with the Asset Framework are:

1. Add the required fields, Asset Framework references, and finders to the model entity.
2. Manage the asset lifecycle (usually in the CRUD methods on the service layer).
  - Whenever you modify the custom model entity, the corresponding asset entry has to be updated as well.
3. Create an asset renderer factory for providing the renderer the model entity.
4. Create an asset renderer for displaying the model entity.

Additionally, if you want to show your assets in the Asset Publisher:

1. Implement the JSP files to support the different display modes of the Asset Publisher.
2. Integrate with the Liferay Search framework (required).

See the Developer Network article:

[https://dev.liferay.com/fi/develop/tutorials/-/knowledge\\_base/7-2/assets-integrating-with-liferays-framework](https://dev.liferay.com/fi/develop/tutorials/-/knowledge_base/7-2/assets-integrating-with-liferays-framework) for more information.

## Integrate with the Search Framework

Before version 7.1, there used to be a single indexer component for taking care of everything search indexer related for a model entity. In 7.1 the approach was modularized to provide a clean approach for controlling different aspects of search framework integration. You can still use the old approach, however.

Generally, the steps to integrate with the Search framework are:

1. Implement a Model Registrar class to register with the search framework
2. Implement a Model Document Contributor to control which fields are indexed
3. Implement a Model Indexer Writer Contributor to configure reindexing
4. Implement a Keyword Query Contributor to control which fields of the model are being queried
5. Implement a Model Summary Contributor to control the summaries returned
6. Add the `@Indexable` annotations to the service methods that should trigger indexing.
7. Integrate with the Asset Framework to be able to show the entities in the Asset Publisher

See details about all the available contributors and integrating to Search Framework in the Developer Network:

[https://dev.liferay.com/en/develop/tutorials/-/knowledge\\_base/7-2/search-and-indexing](https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/search-and-indexing). For a real world example, see  
<https://github.com/liferay/liferay-portal/tree/7.2.x/modules/apps/blogs/blogs-service/src/main/java/com/liferay/blogs/internal/search>.

## Keeping the Search Index Updated on Entity Modification Events

When you use Service Builder, the Liferay-provided method level `@Indexable` annotation automates the task for you. Just annotate those methods, which should trigger an index update. The only requirement for annotated methods is that they **have to return the target entity**.

The `@Indexable` annotation has two action types:

- **REINDEX:** on add or update
- **DELETE:** on entity delete

Below is an example from a Service Builder project implementation class, where an index document gets created on entity add and deleted on deletion:

### addAssignment()

```
@Indexable(
 type = IndexableType.REINDEX
)
public Assignment addAssignment(
 long groupId, Map<Locale, String> titleMap, String description,
 Date dueDate, ServiceContext serviceContext)
throws PortalException {
 ...
 return assignment;
}
```

### deleteAssignment()

```
@Indexable(
 type = IndexableType.DELETE
)
public Assignment deleteAssignment(Assignment assignment)
throws PortalException {
 ...
 return assignment;
}
```

Notice that by default, the CRUD methods of Service Builder generated service **base classes** are annotated with @Indexable automatically. If you call the annotated base class methods from your implementation class, you don't have to annotate them.

## Creating a Custom Search Interface

To get your custom entities to show up in standard portal search, you have to integrate with the Liferay Asset Framework. If, however, you would like to create your own custom search user interface, you can call the extensive portal search API directly.

Generally, the steps to create a custom Liferay search interface are:

1. Send the query parameters from the user interface to the back-end.
2. Catch the parameters and build a *SearchContext* object that transports all the required information to the search engine adapter.
3. Call the *IndexSearcherHelper* service and execute the search.
4. Format the *Hits* objects for the user interface

Below is an example of creating the *SearchContext* object and executing the search:

```
public Hits doSearch(ThemeDisplay themeDisplay, BooleanClause booleanClause, int start, int end, Sort[] sorts) {

 SearchContext searchContext = new SearchContext();
 searchContext.setCompanyId(themeDisplay.getCompanyId());
 searchContext.setStart(start);
 searchContext.setEnd(end);
 searchContext.setSorts(sorts);

 searchContext.setBooleanClauses(new BooleanClause[] {
 booleanClause
 });

 Hits hits = _indexSearcherHelper.search(searchContext, query)
;

 return hits;
}

@Reference
private IndexSearcherHelper _indexSearcherHelper;
```

## Exercises

# 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

### Add the Required Fields to the Assignment Entity

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.

1. Open the `service.xml` in the *gradebook-service* module.
2. Add the status fields (see the highlighted fields). The final `service.xml` will look like this:

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder 7.2.0//EN" "http://www.liferay.com/dtd/liferay-service-builder_7_2_0.dtd">
```

```

<service-builder dependency-injector="ds" package-path="com.liferay.training.gradebook">
 <namespace>Gradebook</namespace>
 <!--<entity data-source="sampleDataSource" local-service="true" name="Foo"
 remote-service="false" session-factory="sampleSessionFactory" table="foo"
 tx-manager="sampleTransactionManager" uuid="true""> -->

 <entity local-service="true" name="Assignment" remote-service="true" uuid="true">

 <!-- PK fields -->

 <column name="assignmentId" primary="true" type="long"></column>

 <!-- 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="title" type="String" localized="true"></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" />

```

```
<!-- 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 -->

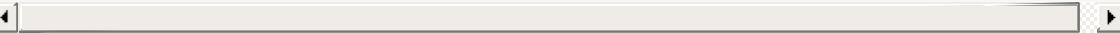
<reference entity="Group" package-path="com.liferay.portal"></reference>

<!-- Entity services needed for the integration to Asset framework -->

<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 {
 ...
}
```

## Implement Asset Resource Management in the Local Service

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:

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.getUuid(), 0, serviceContext.getAssetCategoryIds(),
 serviceContext.getAssetTagNames(), true, true,
 assignment.getCreateDate(), null, null, null,
 ContentTypes.TEXT_HTML,
 assignment.getTitle(serviceContext.getLocale()),
 assignment.getDescription(serviceContext.getLocale())
, 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.

## Implement an Asset Renderer Factory for the Assignments

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*.

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.

Create an AssetRendererFactory component to take care of providing the AssetRenderer:

1. **Create** a class

```

com.liferay.training.gradebook.web.asset.model.AssignmentAssetRe
ndererFactory and implement as follows:

```

```

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

import com.liferay.asset.display.page.portlet.AssetDisplayPag

```

```
eFriendlyURLProvider;
import com.liferay.asset.kernel.model.AssetRenderer;
import com.liferay.asset.kernel.model.AssetRendererFactory;
import com.liferay.asset.kernel.model.BaseAssetRendererFactor
y;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.LiferayPortletReques
t;
import com.liferay.portal.kernel.portlet.LiferayPortletRespon
se;
import com.liferay.portal.kernel.portlet.LiferayPortletURL;
import com.liferay.portal.kernel.portlet.PortletURLFactory;
import com.liferay.portal.kernel.security.permission.ActionKe
ys;
import com.liferay.portal.kernel.security.permission.Permissi
onChecker;
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.GradebookCons
stants;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocal
Service;
import com.liferay.training.gradebook.web.constants.Gradebook
PortletKeys;
import com.liferay.training.gradebook.web.constants.MVCComman
dNames;

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 c
lassTypeId) {

 PortletURL portletURL = _portal.getControlPanelPortle
tURL(
 liferayPortletRequest, getGroup(liferayPortletReq
uest),
 GradebookPortletKeys.Gradebook, 0, 0, PortletRequ
est.RENDER_PHASE);

 portletURL.setParameter("mvcRenderCommandName", MVCCo
mmandNames.EDIT_ASSIGNMENT);

 return portletURL;
}

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

 LiferayPortletURL liferayPortletURL =
 liferayPortletResponse.createLiferayPortletURL(
 GradebookPortletKeys.Gradebook, PortletReques
```

```

t.RENDER_PHASE);

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

 return liferayPortletURL;
}

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

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

}

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

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

@Reference
private AssetDisplayPageFriendlyURLProvider
 _assetDisplayPageFriendlyURLProvider;

@Reference
private AssignmentLocalService _assignmentLocalService;

@Reference(

```

```

 target = "(model.class.name=com.liferay.training.grad
ebook.model.Assignment)"
)
private ModelResourcePermission<Assignment>
 _assignmentModelResourcePermission;

@Reference
private Portal _portal;

@Reference(
 target = "(resource.name=" + GradebookConstants.RESOU
RCE_NAME + ")"
)
private PortletResourcePermission _portletResourcePermiss
ion;

@Reference
private PortletURLFactory _portletURLFactory;

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

}

```

## Implement an AssetRenderer for the Assignments

Create the AssetRenderer class for the Assignments:

1. **Create** a class

```
com.liferay.training.gradebook.web.asset.model.AssignmentAssetRe
nderer
```

and implement as follows:

```

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

import com.liferay.asset.display.page.portlet.AssetDisplayPag
eFriendlyURLProvider;

```

```
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.AssignmentPermission;

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 BaseJSPAssetRend
erer<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, Stri
ng 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 portle
tResponse) {

 ThemeDisplay themeDisplay = (ThemeDisplay)portletRequ
est.getAttribute(
 WebKeys.THEME_DISPLAY);

 int abstractLength = AssetHelper.ASSET_ENTRY_ABSTRACT
_LENGTH;

 String summary = HtmlUtil.stripHtml(
 StringUtil.shorten(
 _assignment.getDescription(themeDisplay.getLo
cale()),
 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, GradebookPortletKey.s.Gradebook, 0, 0,
 PortletRequest.RENDER_PHASE);

 portletURL.setParameter(
 "mvcRenderCommandName", MVCCCommandNames.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(), GradebookPortletKey
 .Gradebook
);
}

PortletURL portletURL;

if (plid == LayoutConstants.DEFAULT_PLID) {
 portletURL = liferayPortletResponse.createLiferayPortletURL(
```

```

 getControlPanelPlid(liferayPortletRequest
),
 GradebookPortletKeys.Gradebook,
 PortletRequest.RENDER_PHASE);
 }
 else {
 portletURL =
 PortletURLFactoryUtil.getPortletURLFactor
y(
).create(
 liferayPortletRequest, GradebookPortl
etKeys.Gradebook,
 plid, PortletRequest.RENDER_PHASE
);
 }

 portletURL.setParameter(
 "mvcRenderCommandName", MVCCCommandNames.VIEW_
ASSIGNMENT);
 portletURL.setParameter(
 "assignmentId", String.valueOf(_assignment.ge
tAssignmentId()));

 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.getUuid();
}

@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 [Liferay Developer Network](#) for more information about Asset renderers.

## Implement the JSP files

At 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 templ
ate) {

 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(WebKeys.ASSET_RENDERER);
%>

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

#### **full\_content.jsp**

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

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

 String viewEntryURL = assetRenderer.getURLView(liferayPortletResponse,WindowState.MAXIMIZED);

 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(locale)) %>
</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.

## Exercises

### 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 [Developer Network](#) for more information. Also, take a look at the optional Module 6 exercise "Enable Workflows for Assignments" where we will cover the `PreFilterContributor`.

## Declare Dependencies

Integration to portal search depends on both the Search API and Search SPI:

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"
```

## Implement a Gradebook Registrar Class

The registrar class registers the Assignments with the search framework:

1. **Create** a class

```
com.liferay.training.gradebook.internal.search.AssignmentSearchR
egistrar
```

 in the *gradebook-service* module.

2. **Implement** as follows: `java package com.liferay.training.gradebook.internal.search;`

```
import com.liferay.portal.kernel.search.Field;
import com.liferay.portal.search.spi.model.index.contributor.Mode
lIndexerWriterContributor;
import com.liferay.portal.search.spi.model.registrar.ModelSearchR
egistrarHelper;
import com.liferay.portal.search.spi.model.result.contributor.Mod
elSummaryContributor;
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.setDefaultValueNames(
 Field.ASSET_TAG_NAMES, Field.COMPANY_ID,
 Field.ENTRY_CLASS_NAME, Field.ENTRY_CLASS_PK,
 Field.GROUP_ID, Field.MODIFIED_DATE, Field.SCOPE_GROUP_ID,
 Field.UID);

 modelSearchDefinition.setDefaultSelectedLocalizedFieldNames(
 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.Assignment)"
)
 protected ModelIndexerWriterContributor<Assignment>
 modelIndexWriterContributor;
}

```

```

 @Reference
 protected ModelSearchRegistrarHelper modelSearchRegistrarHelp
er;

 @Reference(
 target = "(indexer.class.name=com.liferay.training.gradeb
ook.model.Assignment)"
)
 protected ModelSummaryContributor modelSummaryContributor;

 private ServiceRegistration<?> _serviceRegistration;

}
```

```

Implement an Assignment Model Document Contributor

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.

- Create** a class

```
com.liferay.training.gradebook.internal.search.spi.model.index.c
ontributor.AssignmentModelDocumentContributor in the gradebook-service
module.
```

- Implement** as follows:

```

package com.liferay.training.gradebook.internal.search.spi.mo
del.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.grade
book.model.Assignment",
    service = ModelDocumentContributor.class
)
public class AssignmentModelDocumentContributor
    implements ModelDocumentContributor<Assignment> {

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

        // 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.getM
odifiedDate());

        // Handle localized fields.

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

            String languageId = LocaleUtil.toLanguageId(local
e);
```

```

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

```

Implement an Assignment Model Indexer Writer Contributor

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*:

1. **Create** a class

```
com.liferay.training.gradebook.internal.search.spi.model.index.c
ontributor.AssignmentModelIndexerWriterContributor
```

in the *gradebook-service* module.

2. **Implement** as follows:

```

package com.liferay.training.gradebook.internal.search.spi.mo
del.index.contributor;

import com.liferay.portal.kernel.search.Document;
import com.liferay.portal.search.batch.BatchIndexingActionabl
e;
import com.liferay.portal.search.batch.DynamicQueryBatchIndex
ingActionableFactory;
import com.liferay.portal.search.spi.model.index.contributor.
ModelIndexerWriterContributor;
import com.liferay.portal.search.spi.model.index.contributor.
helper.ModelIndexerWriterDocumentHelper;
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(assignment);

                batchIndexingActionable.addDocuments(document);
            });
    }

    @Override
    public BatchIndexingActionable getBatchIndexingActionable() {

        return dynamicQueryBatchIndexingActionableFactory.getBatchIndexingActionable(
            assignmentLocalService.getIndexableActionableDynamicQuery());
    }
}
```

```

}

@Override
public long getCompanyId(Assignment assignment) {

    return assignment.getCompanyId();
}

@Reference
protected AssignmentLocalService assignmentLocalService;

@Reference
protected DynamicQueryBatchIndexingActionableFactory dynamicQueryBatchIndexingActionableFactory;

}

```

Implement a Gradebook Keyword Query Contributor

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

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.KeywordQueryContributorHelper;

```

```

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

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

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

        SearchContext searchContext =
            keywordQueryContributorHelper.getSearchContext();

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

    @Reference
    protected QueryHelper queryHelper;

}

```

Implement a Gradebook Model Summary Contributor

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

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.contributor;

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.ModelSummaryContributor;

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 = 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.DESCRIPTI

```

```

        ON, languageId),
            LocalizationUtil.getLocalizedString(Field.TITLE, languageId));
    }

    private Summary _createSummary(
        Document document, String descriptionField, String titleField) {

        String prefix = Field.SNIPPET + StringPool.UNDERLINE;

        Summary summary = new Summary(
            document.get(prefix + titleField, titleField),
            document.get(prefix + descriptionField, descriptionField));

        summary.setMaxContentLength(200);

        return summary;
    }

}

```

3. Rebuild the service.

Review the Service Implementation Classes for `@Indexable` Annotations

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.

Reindex the Search Index

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

1. **Open Control Panel → Configuration → Search.**
2. **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.

Introducing Liferay's Configuration API

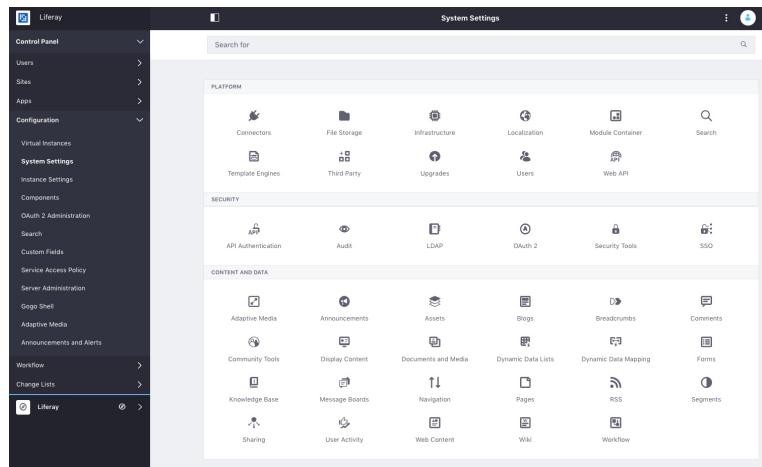
Liferay's configuration API is a configuration management framework for both the Liferay platform and applications on the platform. The configuration API is based on the [OSGi Configuration Admin service](#), which allows you to dynamically set and manage configuration data for OSGi bundles and components. The configuration API is part of the OSGi Compendium specification.

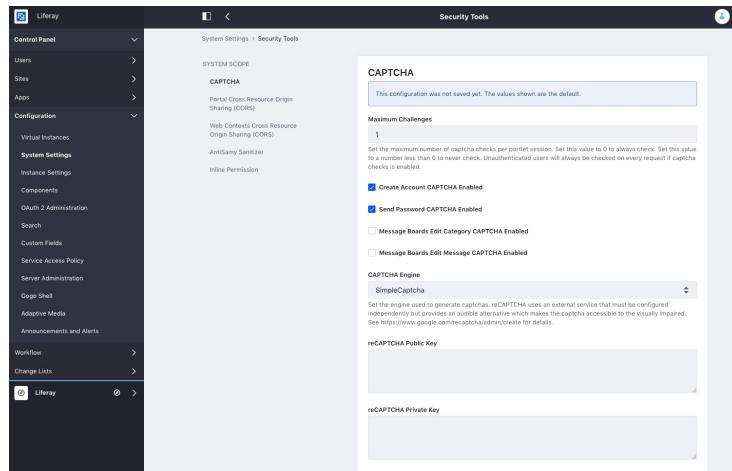
While standard portlet preferences can still be used in portlet applications, the configuration API is more flexible and feature-rich, superseding standard preferences as the preferred portlet application configuration framework.

As with portlet preferences, data in the configuration API is stored in key-value pairs. With the configuration API, however, the values have strong Java typing. Also, similar to portlet preferences on the Liferay platform before, the configuration can be scoped. The available scopes are:

- **System:** unique for the complete system
- **Virtual Instance:** instance-wide
- **Site:** configuration can vary per site
- **Portlet Instance:** single application (portlet) instance

When an application configuration is implemented using the configuration API, a management user interface is generated automatically. The generated management user interface can be accessed in *Control Panel → Configuration → System Settings -> CONFIGURATION_NAME*:





Configuration data can be exported and imported. This is useful, for example, in transporting or replicating settings between environments. Importing configuration data can be done by copying the exported configuration file to the `LIFERAY_HOME/osgi/configs` folder.

On a high level, the minimum steps required for making a Liferay application configurable with the configuration API are as follows:

- If necessary, add required dependencies to the `build.gradle`.
- Create the configuration interface.
- If using other than system scope, create a Configuration Provider
- Make the configuration data available in an OSGi component.

As the Liferay configuration API is relying on the standard OSGi configuration Admin service, the component runtime configuration data can be read with standard OSGi management tools like the Gogo Shell and Felix Web Console.

Example: Creating a System-Wide Portlet Configuration

Step 1 - Add Metatype API Dependency

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

Step 2 - Create the Configuration Interface

Creating the configuration interface automatically creates the configuration user interface in *Control Panel → System Settings*. The configuration `id` property has to match the interface fully qualified name.

```
package com.liferay.training.configuration;

import aQute.bnd.annotation.metatype.Meta;

@Meta.OCD(
    id = "com.liferay.training.configuration.ModuleConfiguration"
    ,
    localization = "content/Language",
    name = "configuration-api-example-portlet",
)
public interface ModuleConfiguration {

    @Meta.AD(
        deflt = "false",
        description = "show-hello-description",
        name = "show-hello-name",
        required = false
    )
    public boolean showHello();

}
```

Step 3 - Make the Configuration Data Available in an OSGi Component

Reference the configuration by its id using the component property `configurationPid`. The configuration variable has to be `volatile`. After that, the configuration can be consumed, for example, by putting values into the request:

```
@Component(
    configurationPid = "com.liferay.training.configuration.ModuleConfiguration",
    immediate = true,
    property = {
```

```
"com.liferay.portlet.display-category=category.sample",
"com.liferay.portlet.instanceable=true",
"javax.portlet.display-name=Make-application-configurable
Portlet",
"javax.portlet.init-param.template-path=/",
"javax.portlet.init-param.view-template=/view.jsp",
"javax.portlet.name=" + ConfigurationExamplePortletKeys.C
ONFIGURATION_EXAMPLE,
"javax.portlet.resource-bundle=content.Language",
"javax.portlet.security-role-ref=power-user,user"
},
service = Portlet.class
)
public class ConfigurationExamplePortlet extends MVCPortlet {

    @Activate
    @Modified
    protected void activate(Map<String, Object> properties) {

        _moduleConfiguration = ConfigurableUtil.createConfigurabl
e(
            ModuleConfiguration.class, properties);
    }

    @Override
    public void render(
        RenderRequest renderRequest, RenderResponse renderRespons
e)
        throws IOException, PortletException {

        renderRequest.setAttribute("showHello", _moduleConfigurat
ion.showHello());

        super.render(renderRequest, renderResponse);
    }

    private volatile ModuleConfiguration _moduleConfiguration;
}
```

A note on the annotations: `@Meta.OCD` and `@Meta.AD` are part of the bnd library, while `@ObjectClassDefinition` and `@AttributeDefinition` are OSGi core equivalents. Both can be used, but only bnd annotations are available at runtime.

Further Reading

- Configurable Applications: https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/configurable-applications

Knowledge Check

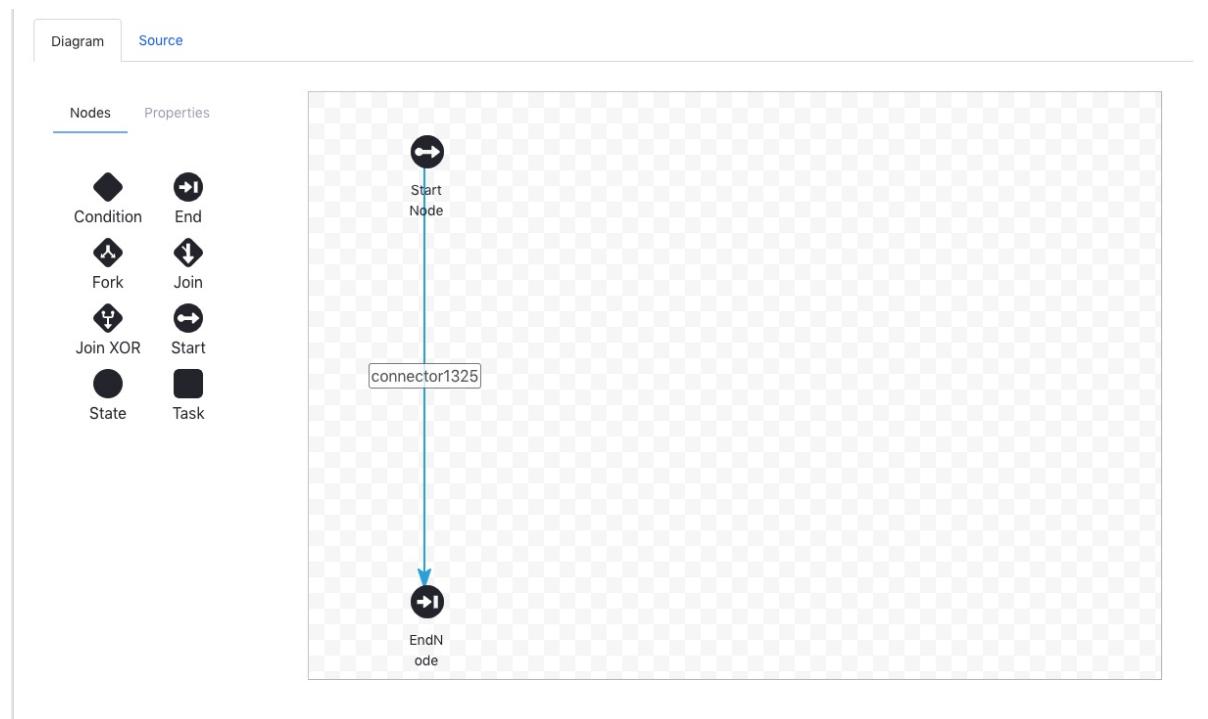
- Liferay's _____ is a _____ for both the Liferay platform and applications.
- The _____ is based on OSGi _____.
- Configuration data is stored in _____ and can be _____.

Implement Workflow Support Introduction

A workflow is an orchestrated and repeatable pattern or sequence of operations. Typically, workflows are used in review processes, but they're often used in integrating with other systems as well.

In Liferay, any registered asset can be assigned to a workflow. For handling the workflows, Liferay is using its own Kaleo workflow engine, but integrations to other workflow engines are available from third parties.

For designing the workflows, there is a workflow editor available in the *Control Panel*.



Implementing Support for Workflows

Here are the general steps for enabling workflows for a custom entity:

1. Ensure that the model entity has status fields.
2. Add workflow instance creation and deletion handling to the service layer.
3. Make getter methods on the service layer status-aware.
4. Add a method on the service layer for updating the status.
5. Create a workflow handler component.
6. Implement the status on the user interface as needed.

Implementation Example

As an example, let's take a look at the workflow support implementation in the Liferay Blogs application: <https://github.com/liferay/liferay-portal/blob/7.1.x/modules/apps/blogs>.

1. Status Field in service.xml

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder
7.1.0//EN" "http://www.liferay.com/dtd/liferay-service-builder_7
_1_0.dtd">

<service-builder auto-import-default-references="false" auto-name-
space-tables="false" package-path="com.liferay.blogs">
    <namespace>Blogs</namespace>
    <entity local-service="true" name="BlogsEntry" remote-service=
"true" trash-enabled="true" uuid="true">
        ...
        <column name="status" type="int" />
        <column name="statusByUserId" type="long" />
        <column name="statusByUserName" type="String" uad-anonymi-
ze-field-name="fullName" />
        <column name="statusDate" type="Date" />
        ...
    </entity>
</service-builder>
```

2. Workflow Instance Creation and Deletion Handling on the Service Layer

Workflow instance creation and deletion is handled in

```
com.liferay.blogs.service.impl.BlogsEntryLocalServiceImpl :
```

Create workflow instance

```
@Indexable(type = IndexableType.REINDEX)
@Override
```

```

public BlogsEntry addEntry(
    long userId, String title, String subtitle, String urlTitle,
    String description, String content, Date displayDate,
    boolean allowPingbacks, boolean allowTrackbacks,
    String[] trackbacks, String coverImageCaption,
    ImageSelector coverImageImageSelector,
    ImageSelector smallImageImageSelector,
    ServiceContext serviceContext)
throws PortalException {
    ...
    return startWorkflowInstance(userId, entry, serviceContext);
}

```

Delete workflow instance

```

@Indexable(type = IndexableType.DELETE)
@Override
@SystemEvent(type = SystemEventConstants.TYPE_DELETE)
public BlogsEntry deleteEntry(BlogsEntry entry) throws PortalException {
    ...
    workflowInstanceLinkLocalService.deleteWorkflowInstanceLinks(
        entry.getCompanyId(), entry.getGroupId(),
        BlogsEntry.class.getName(), entry.getEntryId());
    ...
    return entry;
}

```

3. Status-Aware Getter Methods

Status in a parameter in the getter methods in

```
com.liferay.blogs.service.impl.BlogsEntryLocalServiceImpl :
```

```

@Override
public List<BlogsEntry> getCompanyEntries(
    long companyId, Date displayDate,
    QueryDefinition<BlogsEntry> queryDefinition) {

    if (queryDefinition.isExcludeStatus()) {
        return blogsEntryPersistence.findByC_LtD_NotS(
            companyId, displayDate, queryDefinition.getStatus
        ),
            queryDefinition.getStart(), queryDefinition.getEn
        d(),
            queryDefinition.getOrderByComparator());
    }
    else {
        return blogsEntryPersistence.findByC_LtD_S(
            companyId, displayDate, queryDefinition.getStatus
        ),
            queryDefinition.getStart(), queryDefinition.getEn
        d(),
            queryDefinition.getOrderByComparator());
    }
}
...

```

4. Method for Updating Status

The status updating method is in the

```
com.liferay.blogs.service.impl.BlogsEntryLocalServiceImpl :
```

```

@Indexable(type = IndexableType.REINDEX)
@Override
public BlogsEntry updateStatus(
    long userId, long entryId, int status,
    ServiceContext serviceContext,
    Map<String, Serializable> workflowContext)
throws PortalException {
    // Entry
    User user = userLocalService.getUser(userId);
    Date now = new Date();

```

```

        BlogsEntry entry = blogsEntryPersistence.findByPrimaryKey
(entryId);

        validate(
            entry.getTitle(), entry.getUrlTitle(), entry.getConte
nt(), status);

        int oldStatus = entry.getStatus();

        if ((status == WorkflowConstants.STATUS_APPROVED) &&
            now.before(entry.getDisplayDate())) {

            status = WorkflowConstants.STATUS_SCHEDULED;
        }

        ...

        return entry;
    }
}

```

5. Workflow Handler Component

Workflow handler component

```
com.liferay.blogs.internal.workflow.BlogsEntryWorkflowHandler :
```

```

@Component(
    property = "model.class.name=com.liferay.blogs.model.BlogsEnt
ry",
    service = WorkflowHandler.class
)
public class BlogsEntryWorkflowHandler extends BaseWorkflowHandler
<BlogsEntry> {

    ...

    @Override
    public BlogsEntry updateStatus(
        int status, Map<String, Serializable> workflowContext)
}

```

```

        throws PortalException {

    long userId = GetterUtil.getLong(
        (String)workflowContext.get(WorkflowConstants.CONTEXT
_CONTEXT_USER_ID));
    long classPK = GetterUtil.getLong(
        (String)workflowContext.get(
            WorkflowConstants.CONTEXT_ENTRY_CLASS_PK));

    ServiceContext serviceContext = (ServiceContext)workflowC
ontext.get(
    "serviceContext");

    return _blogsEntryLocalService.updateStatus(
        userId, classPK, status, serviceContext, workflowCont
ext);
}

```

...

6. Implementation on the User Interface

Status field is displayed, for example, in the Blogs search page:

```

<c:choose>
    <c:when test='<%= displayStyle.equals("descriptive") %>'>
        <liferay-ui:search-container-column-user
            showDetails="<%= false %>"
            userId="<%= entry.getUserId() %>">
        />

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

        <%
            Date modifiedDate = entry.getModifiedDate();

```

```
        String modifiedDateDescription = LanguageUtil.getTime
Description(request,
        System.currentTimeMillis() - modifiedDate.getTime(),
true);
    %>

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

    <h4>
        <aui:a href="<%=
rowURL.toString() %>">
            <%= BlogsEntryUtil.getDisplayTitle(resourceBu
ndle, entry) %>
        </aui:a>
    </h4>

    <h5 class="text-default">
        <aui:workflow-status markupView="lexicon" showIcon
="<%=
false %>" showLabel="<%=
false %>" status="<%=
entry.getSta
tus() %>" />
    </h5>
</liferay-ui:search-container-column-text>

<liferay-ui:search-container-column-jsp
    path="/blogs_admin/entry_action.jsp"
    />
</c:when>
</c:choose>
```



Integrate with External Systems

Liferay Service Builder can automatically generate JSON and SOAP web services APIs for your service. As Liferay platform core services are created using the Service Builder pattern, they all have web service APIs available.

In addition to Service Builder generated web services, it is possible to publish JAX-RS REST and JAX-WS endpoints for any ad-hoc service.

Enabling Remote Services

Setting the `remote-service` attribute to true in the Service Builder entity definition creates the remote service variant for the entity and adds the `@JSONWebService` annotation to the service interface, making all the public methods of that interface available as JSON web services:

service.xml

```
<entity name="Assignment" uuid="true" local-service="true" remote-service="true">
```

AssignmentService.java

```
@AccessControlled
@JSONWebService
@OSGiBeanProperties(
    property = {
        "json.web.service.context.name=gradebook",
        "json.web.service.context.path=Assignment"
    },
    service = AssignmentService.class
)
@ProviderType
@Transactional(
    isolation = Isolation.PORTAL,
    rollbackFor = {
        PortalException.class, SystemException.class
}
```

```
    }
)
public interface AssignmentService extends BaseService {
    ...
}
```

Ignoring a Method

A method can be prevented from being exposed as a web service by setting the mode attribute to `JSONWebServiceMode.IGNORE` in the remote service implementation class:

```
@JSONWebService(
    mode = JSONWebServiceMode.IGNORE
)
public String myIgnoredMethod() {
```

Defining HTTP Methods

JSON enabled services are mapped to GET or POST HTTP methods with the following logic:

- **GET:** if method name starts with "get", "is", or "has"
- **POST:** all other method prefixes

HTTP methods can, however, be explicitly defined on a method level by setting the method attribute:

```
@JSONWebService(
    value = "do-some-thing",
    method = "PUT"
)
public void doSomething() {
```

Explicit Method Registration

By setting the JSONWebService mode to `MANUAL`, the methods to be exposed can be declared manually. In the example below, only the `getAssignment()` method is exposed to the web service API.

```
@JSONWebService(  
    mode = JSONWebServiceMode.MANUAL  
)  
public class AssignmentServiceImpl extends AssignmentServiceBaseI  
mpl{  
  
    @JSONWebService  
    public Assignment getAssignment(long assignmentId) {  
        ...  
    }  
  
    public void addAssignment(Assignment assignment) {  
        ...  
    }  
}
```

Configuring the JSON Web Service API Properties

Global JSON Web Service configuration is done in the [portal properties](#). For example, the following settings are available:

```
# Enable / disable JSON web service API  
json.web.service.enabled=false  
  
# Discoverability through the test page http://[address]:[port]/api/jsonws  
jsonws.web.service.api.discoverable=false  
  
# Restricted HTTP methods  
jsonws.web.service.invalid.http.methods=DELETE,POST,PUT  
  
# By default, the HTTP method is not checked when invoking a service call. This setting enables the strict mode.  
jsonws.web.service.strict.http.method=true
```

```
# Web service paths that are accessible
jsonws.web.service.paths.includes=get*,has*,is*,  
  

# Web service paths that aren't allowed. This setting takes precedence over the jsonws.web.service.paths.includes
jsonws.web.service.paths.excludes=set*, add*
```

Testing the JSON Web Service API

A JSON web service test page can be accessed at <http://localhost:8080/api/jsonws>. URL, cURL, and JavaScript examples are provided:

JSONWS API

Context Name
HTTP Method **GET**

space
Search

/space.assignment/get-assignments-by-group-id

com.liferay.training.space.gradebook.service.impl.**AssignmentServiceImpl**

getAssignmentsByGroupId

Parameters

p_auth String authentication token used to validate the request

groupId long

start int

end int

Return Type

java.util.List

Exception

Execute

p_auth	XaKaj7vt	String
groupId		long
Start		int
End		int

Invoke

Service Builder and SOAP API

Liferay uses [Apache Axis](#) for the SOAP services. To generate the SOAP API for your custom application, the WSDD builder task has to be added to the project to create the web service definition. The WSDD creation process is described in detail in this [Liferay Developer Network article](#).

Configuring the SOAP API Properties

The hosts allowed to access the SOAP API can be defined explicitly in the portal-ext.properties:

```
axis.servlet.hosts.allowed=192.168.100.100, 127.0.0.1, [SERVER_IP]  
]
```

Testing the SOAP API

A test page is available at <http://localhost:8080/api/axis>

The screenshot shows a Mozilla Firefox browser window with the URL localhost:8080/api/axis in the address bar. The page content is titled "And now... Some Services" and lists several service endpoints with their corresponding WSDL links:

- Portlet_ExportImport_ExportImportService ([wsdl](#))
 - exportLayoutsAsFileInBackground
 - exportLayoutsAsFileInBackground
 - exportPortletInfoAsFileInBackground
- Portlet_Announcements_AnnouncementsDeliveryService ([wsdl](#))
 - updateDelivery
 - updateDelivery
- Portal_OrgLaborService ([wsdl](#))
 - addOrgLabor
 - deleteOrgLabor
 - getOrgLabor
 - getOrgLabors
 - updateOrgLabor
- Portal_ImageService ([wsdl](#))
 - getImage
- Portal_PortletPreferencesService ([wsdl](#))
 - deleteArchivedPreferences
- Portlet_DL_DLTrashService ([wsdl](#))
 - moveFileEntryFromTrash
 - moveFileEntryToTrash
 - moveFolderFromTrash
 - moveFolderToTrash
 - restoreFileEntryFromTrash
 - restoreFileShortcutFromTrash
 - restoreFolderFromTrash
- Portlet_DL_DLAppService ([wsdl](#))
 - addFileEntry
 - addFolder
 - cancelCheckOut
 - checkInFileEntry
 - checkInFileEntry
 - checkOutFileEntry
 - checkOutFileEntry
 - copyFolder
 - deleteFileEntryByTitle
 - deleteFileEntry

Publishing JAX-RS and JAX-WS Services

Liferay supports publishing JAX-WS and JAX-RS services via the Apache CXF implementation. Publishing JAX-WS and JAX-RS services requires defining an endpoint and an extender. CXF endpoints are context paths the JAX web services are deployed to and accessible from. Extenders specify where the services are deployed:

- **SOAP Extenders:** for publishing JAX-WS web services. Each SOAP extender can deploy the services to one or more CXF endpoints.
- **REST Extenders:** for publishing JAX-RS web services. REST extenders for JAX-RS services are analogous to SOAP extenders for JAX-WS services.

Steps for Publishing a JAX Web Service:

1. Create an OSGi service component for the web service.
2. Configure a Liferay endpoint to access the REST service.
3. Map the endpoint to your REST service using the REST or SOAP extender.

| See [Developer Network](#) for more information.

REST Builder

Liferay 7.2 provides a new API generator tool which consumes OpenAPI profiles and generates the API scaffolding: JAX-RS endpoints, parsing, XML generation, and advanced features like filtering or multipart (binary files) support. The developer only has to fill the resource implementations, calling liferay remote services.

| The [Rest Builder](#) source code.

Knowledge Check

- In addition to Service Builder-generated web services, it is possible to publish

for any

.

Logging

Logging is a powerful method for emitting application metrics and statistics and for helping with troubleshooting and resolving production issues. When properly used, logging improves application quality and maintainability. Examples of use cases:

- Adding additional, contextual information like user ID or thread ID to stack traces
- Collecting and sending metrics like execution timers
- Emitting a warning when a threshold, for example, for group count, is exceeded

Although the role of logging in troubleshooting can be essential, in many cases, logging doesn't expose the root cause, but just detects a problem. Debugging helps find the underlying problem.

Overview

The following out-of-the-box options are available to implement logging in your custom modules:

- Java native logger
- Liferay native logger
- [SLF4J](#) logger
- OSGi Log Service

Except for the Java native logger, all the options use SLF4J and [Log4J](#) as an implementation in the background. Using Java native logging is usually not recommended because of its limitations and performance compared to SLF4J implementations.

Let's take a look at the different ways to invoke logging.

Using Liferay Native Logger

Liferay native logger is called via `LogFactoryUtil.getLog(CLASS_NAME)`.

```
package com.liferay.training.sample.log;

import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
```

```
import java.math.BigDecimal;

public class EmployeeHandler {

    public void setSalary(BigDecimal salary) {

        if (_log.isInfoEnabled()) {
            if (salary.compareTo(MAX_SALARY) == 1) {
                _log.info("Alert: suspiciously high salary: " +
salary + ".");
            }
        }

        _salary = salary;
    }

    private BigDecimal _salary;

    public static final BigDecimal MAX_SALARY;

    static {
        MAX_SALARY = new BigDecimal("10000.0");
    }

    private static final Log _log = LogFactoryUtil.getLog(EmployeeHandler.class);
}

}
```

Using SLF4J Logger

In this approach, you'd first declare the dependency for the SLF4J API. The logger could then be invoked by calling the SLF4J `LoggerFactory`, as in the example below. This logging approach is the recommended one, as it doesn't bind the code to any specific implementation:

build.gradle

```
compileOnly group: "org.slf4j", name: "slf4j-api", version: "1.7.2"
```

Java class

```
package com.liferay.training.sample.log;

import java.math.BigDecimal;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class EmployeeHandler {

    public void setSalary(BigDecimal salary) {

        if (_log.isInfoEnabled()) {
            if (salary.compareTo(MAX_SALARY) == 1) {
                _log.info("Alert: suspiciously high salary: " +
salary + ".");
            }
        }

        _salary = salary;
    }

    private BigDecimal _salary;

    public static final BigDecimal MAX_SALARY;

    static {
        MAX_SALARY = new BigDecimal(10000.0);
    }

    private static final Logger _log = LoggerFactory.getLogger(EmployeeHandler.class);

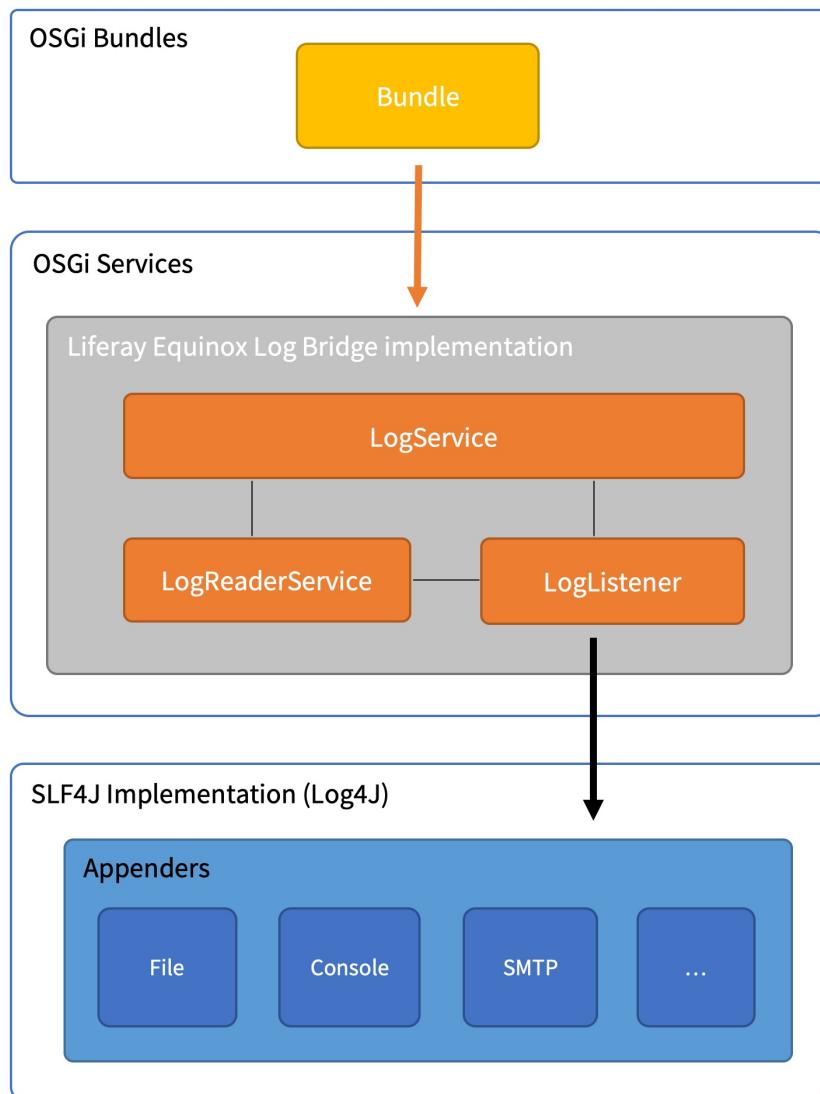
}
```

OSGi Log Service

OSGi Log Service is a message logger service for the OSGi framework, specified in the OSGi Compendium. OSGi Log Service has three main components:

- **LogService**: service interface for storing logs
- **LogReaderService**: service interface for reading and dispatching log entries
- **LogListener**: interface for the listener of log entry objects

In Liferay's Equinox Log Bridge implementation, there's an SLF4J log listener implementation using Log4J as the back-end:



OSGi LogService can be referenced with Declarative Services via the OSGi `@Reference` annotation:

```

package com.liferay.training.sample.log;

import java.math.BigDecimal;

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

@Component(
    service = EmployeeHandlerService.class
)
public class EmployeeHandlerServiceImpl implements EmployeeHandlerService {

    @Override
    public void setSalary(BigDecimal salary) {
        if (salary.compareTo(MAX_SALARY) == 1) {
            _log.log(LogService.LOG_INFO, "Alert: suspiciously high salary: " + salary + ".");
        }
    }

    private BigDecimal _salary;

    public static final BigDecimal MAX_SALARY;

    static {
        MAX_SALARY = new BigDecimal(10000.0);
    }

    @Reference
    private LogService _log;
}

```

For an OSGi LogService to start Liferay logging, there has to be a level definition. There are two ways to define this:

- `MODULE_ROOT/src/main/META-INF/module-log4j.xml`
- *Control Panel → Configuration → Server Administration → Log Levels*

It should be noticed that this logging approach is limited to OSGi components.

Configuring Logging

Logging levels can be globally configured platform-wide from the Control Panel. Changes made there won't, however, persist after restart:

Persistent level settings as well as all the other LOG4J configuration settings like appenders are set with `LIFERAY_WEBAPP_ROOT/WEB-INF/classes/META-INF/portal-log4j-ext.xml`. In the example below, the level for the `com.liferay.blogs` is set to `INFO`:

```
<?xml version="1.0"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j
/">

    <appender name="FILE" class="org.apache.log4j.rolling.Rolling
FileAppender">
        <rollingPolicy class="org.apache.log4j.rolling.TimeBasedR
ollingPolicy">
            <param name="FileNamePattern" value="@liferay.home@/1
ogs/sql.%d{yyyy-MM-dd}.log" />
        </rollingPolicy>
```

```

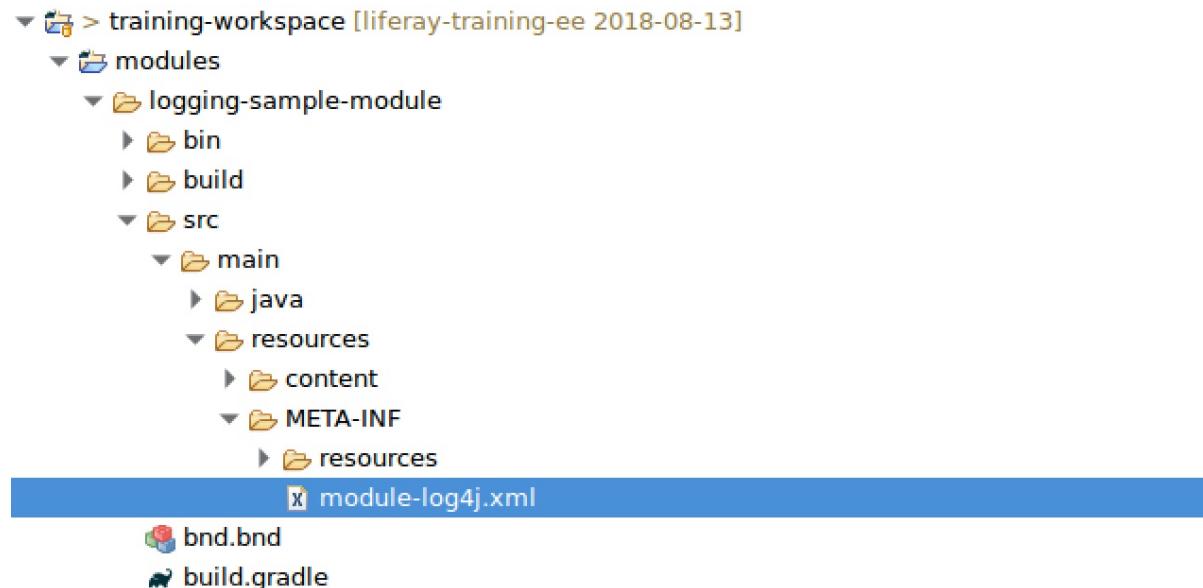
<layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{ABSOLUTE} %
-5p [%c{1}:%L] %m%n" />
</layout>
</appender>

<logger name="com.liferay.blogs">
    <priority value="INFO" />
    <appender-ref ref="FILE" />
</logger>

</log4j:configuration>

```

On a module level, you can define settings for a single module or multiple modules within a module in `MODULE_ROOT/src/main/resources/META-INF/module-log4j.xml` or outside the module , in `LIFERAY_HOME/osgi/log4j/BUNDLE_SYMBOLIC_NAME-log4j-ext.xml` .



In case of fragment bundles, levels for the host bundle can be defined in the fragment bundle's `MODULE_HOME/META-INF/module-log4j-ext.xml` .

Below is an example of the `module-log4j.xml` configuration file. Note the package-naming syntax difference when configuring the OSGi Log Service and native or SLF4J loggers: the OSGi Log Service-related categories have to begin with `osgi.logging` ,

and the dots in the package name have to be replaced with underscores. This same syntax applies when configuring through the *Control Panel*, too:

```
<?xml version="1.0"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j
/>

<!-- OSGi Log Service logger configuration -->

<category name="osgi.logging.com_liferay_training_sample.logs
ervice">
    <priority value="INFO" />
</category>

<!-- Native or SLF4J API logger configuration -->

<category name="com.liferay.training.sample.">
    <priority value="INFO" />
</category>

</log4j:configuration>
```

Other Logging-Related Settings

Although SQL logging is not directly related to Liferay's platform, you can enable logging for Hibernate queries in portal-ext.properties with the following setting:

```
hibernate.show_sql=true
```

JavaScript logging can accordingly be enabled with:

```
javascript.log.enabled=true
```

Remember to disable Hibernate logging in production systems as it produces excessive amount of data.

Wrapping it Up

If you choose to use platform-provided logging functionalities, using SLF4J Logger is recommended for the best code portability.

The logging approach should always be designed and standardized properly. A misuse of logging severities, not readable log messages, or missing contextual information do not necessarily bring any extra value or quality to the application. Properly designed logging can, on the other hand, make applications much easier to maintain and dramatically help with troubleshooting issues.

Too extensive and excessive logging severely impairs application performance. As a rule of thumb, remember also to check if a logging level is enabled before calling the logger, because the clause inside the log call is otherwise evaluated and may impact performance:

```
if (_log.isDebugEnabled()) {  
    _log.debug(DO_AN_EXPENSIVE_FUNCTION_CALL);  
}
```

Knowledge Check

- _____ is a powerful method for emitting _____ and for helping with _____ production issues.
- For the best code portability, using _____ logger is recommended.
- Too extensive and excessive logging severely _____ application performance.

Testing

In this section, we'll discuss some general principles, practices, and guidelines of software testing. The practical part of the section will focus on integration testing.

Why Test?

Humans make mistakes and create bugs in the software. But that's not the only reason for software testing. Here are some additional reasons for testing:

- Finding bugs and defects in the software (functional testing)
- Verifying interaction between software components (integration testing)
- Checking if both functional and non-functional requirements are met (system testing)
- Checking if user acceptance criteria is met (user acceptance testing)
- Managing the development lifecycle (regression testing)

Testing provides concrete metrics about the state of your software. When done properly, it inevitably improves software quality and, ultimately, customer satisfaction.

What You Should Test

Things that generally should be tested include:

- The "business logic"
- CRUD functionalities
- Code or functionalities that are intensively used everywhere in the application
- Code or functionalities developed by multiple developers
- Code or functionality that changes often

Consider quantity vs. quality: having fewer well-targeted tests of good quality is usually better than the opposite.

Functional vs Non-Functional Testing

Software testing is often divided into two main categories:

1. Functional testing
2. Non-functional testing

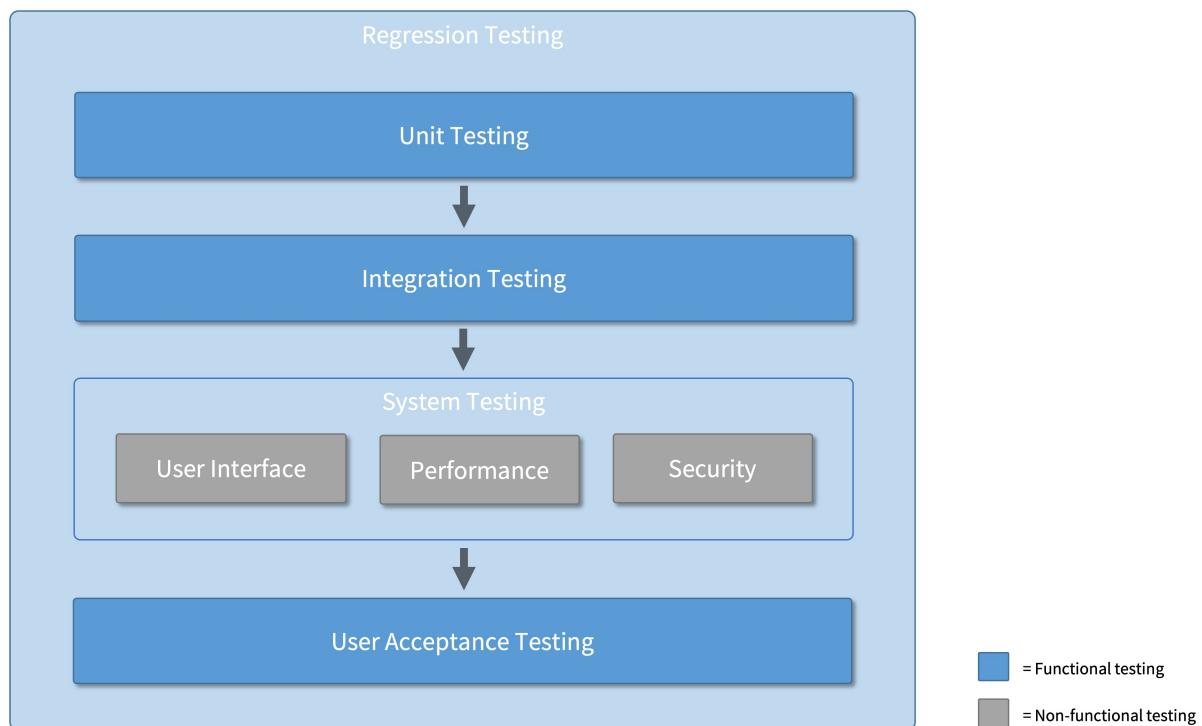
The purpose of **functional testing** is to verify *what* the system does against known requirements. We give tests some input and expect certain kinds of output. Examples of functional testing are unit and integration testing.

Non-Functional Testing is used to verify the way the system works. Non-functional testing checks that all of the components are interacting as designed, the system as a whole works as defined within agreed performance and capacity levels, and the system is ready for production. Examples of this type of testing are, for example, performance and security testing.

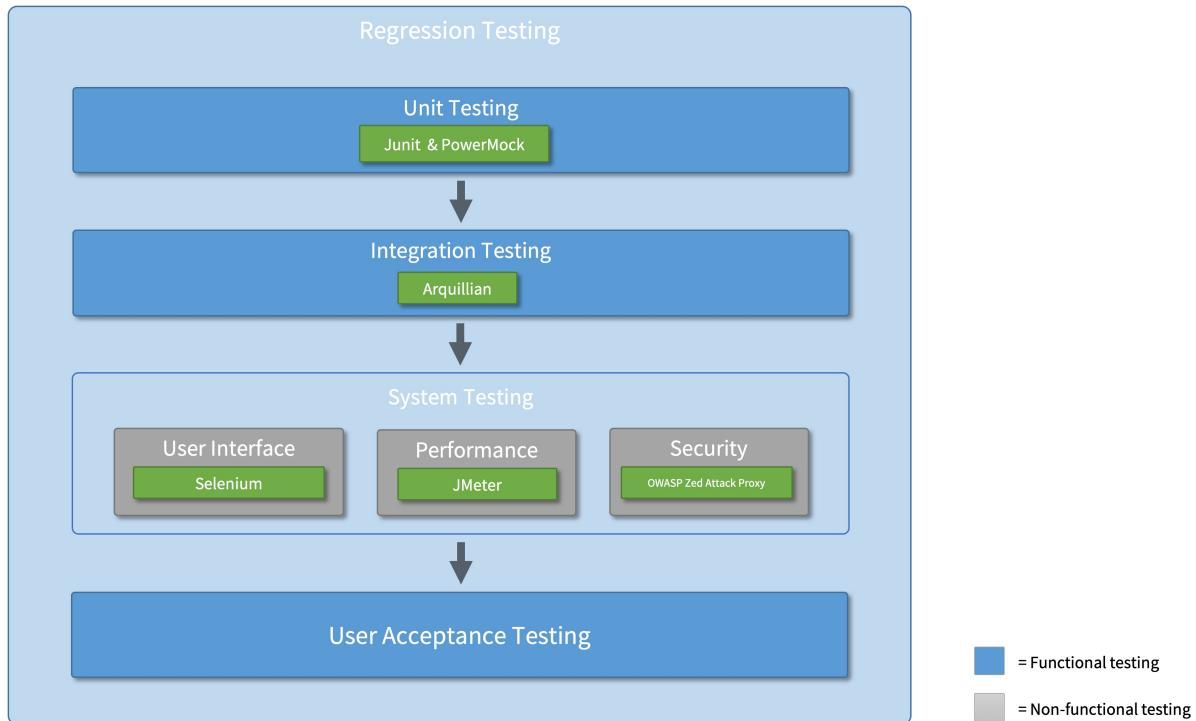
Both functional and non-functional testing categories are necessary. Although functional testing often takes the major role in testing, performance and load tests can expose defects that otherwise would pass the process and go to production. Fixing issues in production is often more difficult, and more expensive.

Stages of Testing

Properly done, testing happens in multiple stages. The smallest software components like methods are tested first, then their interaction, then the complete system and, last, the user perception.



The diagram below shows some tooling examples for the different stages:



General Recommendations for Testing

Make a proper **testing plan** and try to test on all the stages of the development cycle.

A good plan takes care of **regression**. Especially in the beginning of the application lifecycle, there are typically lots of changes in the code, making it prone to regression.

It's practically not possible to cover everything and every possible use case. **Focus on the critical parts** first, and then extend the coverage, but not at the expense of quality. Rely on automation tools.

Test **borderline cases**. Boundary values when it comes to input fields on the user interface or calculations are a popular living area of bugs. These are also the areas where certain kinds of security breaches happen.

Keep tests **simple**. Creating tests can be time-consuming. Try to keep tests simple so that they provide exactly the information they should to remain maintainable over code changes and lifecycles. If your tests seem to be too complex, it might be a sign of too-complex code in the application itself.

Testing Types

Let's do a brief overview of some of the most common functional testing types.

Unit Testing

Unit testing is the first stage of testing. It's meant for the smallest units of code, usually done against methods. Unit tests should be atomic and have only minimal dependencies on other units of code.

A good unit test should:

- Test an isolated component only, with no integrations
- Be able to fail. If a test cannot fail, it's useless.
- Have only one unambiguous reason to fail (often hard to reach)
- Not affect the data other test cases rely on
- Not rely on other tests
- Test just one thing and test it well
- Not have any conditional logic
- Be reliable
- Be repeatable
- Be platform-independent
- Have self-documenting method names

Liferay uses the JUnit framework for unit testing. Below is a simple example of a JUnit test:

```
public class SampleTest {  
  
    @Before  
    public void setUp() {  
  
        _car = new Car();  
    }  
  
    @Test  
    public void testStartCar() {  
  
        _car.start();  
  
        assertEquals(true, car.isEngineRunning());  
    }  
}
```

```
    }

    protected Car _car;
}
```

It's often not possible to isolate code under a test from its dependent components. **Mocking** is an approach to "fake" those dependent components and help to keep the tested code isolated.

Mocking often increases test code complexity and decreases maintainability. In such cases, you should consider using integration testing instead.

Below is an example of a JUnit test using the PowerMock Mockito extension:

```
@PrepareForTest(
{
    CarService.class
}
)
@RunWith(PowerMockRunner.class)

public class SampleTest {

    @Before
    public void setUp() {

        when(carService.getOilLevel(Car.class)).thenReturn(4)
    }

    @Test
    public void testStartCar() {

        Car car = new Car();

        assertTrue(carService.getOilLevel(car) > 0);
    }

    @Mock
```

```
    CarService carService;  
}
```

Integration Testing

In integration testing, a component is tested as a whole with all its integrations to other modules and services. Integration testing tries to expose defects in interfaces and interaction between integrated components.

Liferay uses the [Arquillian Framework](#) and a tailored [Arquillian extension](#) for integration testing. Below is an example of what an Arquillian test could look like:

```
@RunWith(Arquillian.class)  
public class BlogsEntryLocalServiceImplTest {  
  
    @ClassRule  
    @Rule  
    public static final AggregateTestRule aggregateTestRule =  
        new LiferayIntegrationTestRule();  
  
    @Test  
    public void testAddDiscussion() throws Exception {  
        ServiceContext serviceContext =  
            ServiceContextTestUtil.getServiceContext();  
  
        BlogsEntry blogsEntry = BlogsEntryLocalServiceUtil.addEnt  
ry(  
            TestPropsValues.getUserId(), StringUtil.randomString()  
,  
            StringUtil.randomString(), new Date(), serviceContext  
);  
  
        _blogsEntries.add(blogsEntry);  
  
        long initialCommentsCount = CommentManagerUtil.getComment  
sCount(  
            BlogsEntry.class.getName(), blogsEntry.getEntryId());  
  
        CommentManagerUtil.addComment(  
    }
```

```

        TestPropsValues.getUserId(), TestPropsValues.getGroupId(),
        BlogsEntry.class.getName(), blogsEntry.getEntryId(),
        StringUtil.randomString(),
        new IdentityServiceContextFunction(serviceContext));

    Assert.assertEquals(
        initialCommentsCount + 1,
        CommentManagerUtil.getCommentsCount(
            BlogsEntry.class.getName(), blogsEntry.getEntryId
        ));
}
}

```

End-to-End Testing

End-to-end testing, sometimes also called user interface or browser testing, strives to emulate and test the (human) interaction with an application. Probably the most common testing framework in this category is [Selenium](#).

Selenium's core component WebDriver emulates browsers and is also behind [Arquillian's Graphene extension](#). Below is an example using Arquillian Graphene for Liferay portlet user interface testing. The `testInstallPortlet()` gets the current webpage's source and checks if the portlet is found. The other test, `testAdd()`, tests the simple calculator portlet by setting form values:

```

@RunAsClient
@RunWith(Arquillian.class)
public class BasicPortletFunctionalTest {

    @Deployment
    public static JavaArchive create() throws Exception {
        ...
    }

    @Test
    public void testAdd()
        throws InterruptedException, IOException, PortalException
    {

```

```
_browser.get(_portletURL.toExternalForm());
_firstParameter.clear();
_firstParameter.sendKeys("2");
_secondParameter.clear();
_secondParameter.sendKeys("3");

_add.click();
Thread.sleep(5000);
Assert.assertEquals("5", _result.getText());
}

@Test
public void testInstallPortlet() throws IOException, PortalException {
    _browser.get(_portletURL.toExternalForm());

    final String bodyText = _browser.getPageSource();

    Assert.assertTrue("The portlet is not well deployed", bodyText.contains("Sample Portlet is working!"));
}

@FindBy(css = "button[type=submit]")
private WebElement _add;

@Drone
private WebDriver _browser;

@FindBy(css = "input[id$='firstParameter']")
private WebElement _firstParameter;

@PortalURL("arquillian_sample_portlet")
private URL _portletURL;

@FindBy(css = "span[class='result']")
private WebElement _result;

@FindBy(css = "input[id$='secondParameter']")
private WebElement _secondParameter;
```

```
}
```

```
1
```

```
>
```

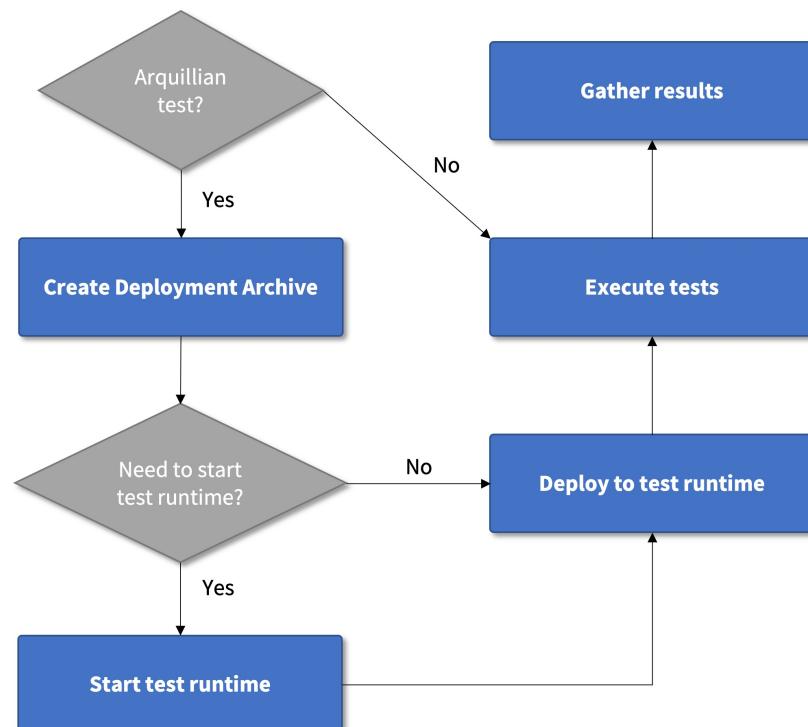
Introducing Arquillian

Arquillian is an integration and functional testing platform for Java. Running tests against a dedicated container, it provides a way to test the Liferay modules comprehensively in a real environment and with real dependencies.

Arquillian can:

- Start and stop the test container
- Create a deployable test archive
- Deploy and un-deploy the test archive
- Enrich the test case with dependency injections
- Execute tests inside the container
- Capture the results

Below is a diagram of an Arquillian test:



Arquillian Components

The Arquillian framework has four main components:

- Test runners
- Containers
- Test enrichers
- Run modes

The Arquillian **Test Runner** extends the JUnit Runner class and takes care of running the Arquillian-decorated test class. The Runner is set with the `@RunWith` annotation:

```
@RunWith(Arquillian.class)
public class AssignmentLocalServiceTest {
    ...
}
```

The **test container** is the server runtime where tests are run. The test container should not be referenced directly in the tests. Container types supported by Arquillian are:

- Remote
- Managed
- Embedded

The embedded container is not supported by the Liferay Arquillian extension.

Test Enrichers are injection resources or service injections into Arquillian test classes:

- `@Resource` - Java EE resource injections
- `@EJB` - EJB session bean reference injections
- CDI-supported injections

If you inject Liferay service references in your test classes, just use `@Inject` instead of `@Reference`:

```
@Inject
private AssignmentLocalService _assignmentLocalService
```

Note that using the `@Inject` annotation requires using the portal integration test framework's [LiferayIntegrationTestRule](#).

Run Mode defines the run context for the test:

- Container mode (default):
 - Deploy the test to the test container.
 - Execute tests in the test container.
- Client mode:
 - Deploy required components to the test container.
 - Execute tests as a client, outside the container.
 - Suitable for testing web services or user interfaces
 - Is set with the `@RunAsClient` annotation

An example of the Arquillian test class in client-run mode:

```
@RunAsClient  
@RunWith (Arquillian.class)  
public class GradebookPortletTest {  
    ...  
}
```

Liferay Arquillian Extension JUnit Bridge

The Arquillian Extension is an extension for Liferay OSGi in-container deployments that takes care of managing the test container, deploying the test bundle, and running the tests.

Steps to Implement Liferay Arquillian Integration Tests

Typical steps for setting up Liferay Arquillian tests:

1. Set up the Liferay Workspace Tomcat bundle.
2. Create a Liferay testing module.
3. Declare testing dependencies, using the `testIntegrationCompile` and `testIntegrationRuntime` scopes.
4. Create the test cases.
5. Run tests using the Gradle `testIntegration` task.

Knowledge Check

- Testing provides _____ about the state of your software.
- Software testing is often divided into two main categories:
 - _____
 - _____
- Testing is done in multiple stages:
 - _____
 - _____
 - _____
 - _____

Debugging

Your application might be failing integration tests or crashing in production because of unknown reasons. The code complexity might have gotten to the point where you aren't sure how everything is working together. Developers make mistakes, and tests can't cover every possible use case scenario. This is where debugging comes in. Debugging covers a number of different approaches, tools, and methodologies under its umbrella.

Here, for the sake of training, we limit the scope to the basics of using the IDE debugger tool, which enables you to monitor and control the execution of a program, setting breakpoints and changing values in memory. We'll also briefly discuss some methods of troubleshooting and debugging issues in production, where there's no source code or IDE available.

Introducing JPDA

JPDA (Java Platform Debugger Architecture) is the backbone of Java debugging. It's a collection of Java APIs to facilitate debugging, profiling, and monitoring local and remote Java applications.

JPDA consists of three layers: JDI, JDWP, and JVM TI.

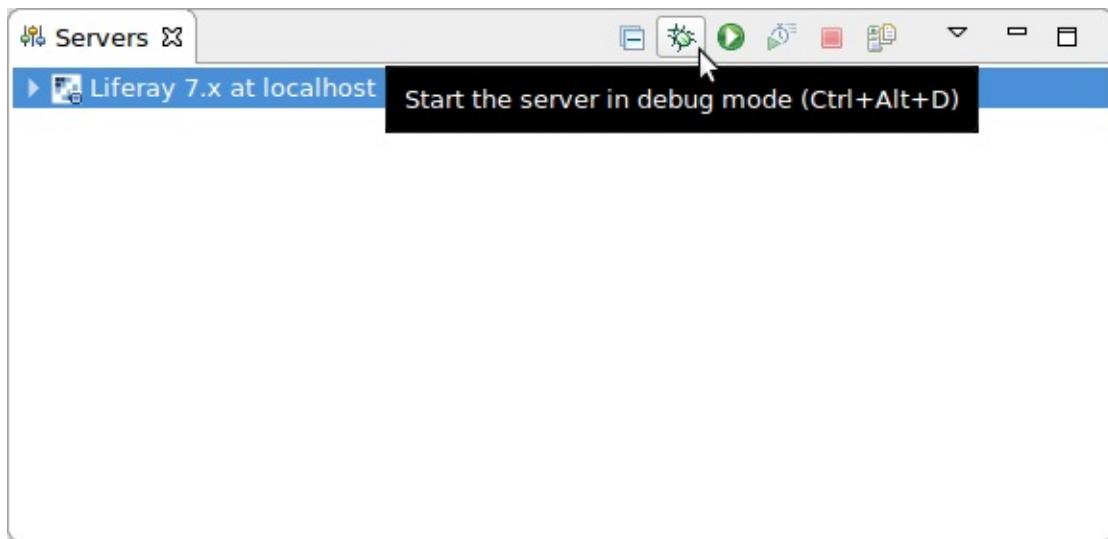
- **JDI** (Java Debug Interface) is the user interface definition layer. In our training context, the implementation is Eclipse-based Dev Studio.
- **JDWP** (Java Debug Wire Protocol) defines the communication between the debuggee and the debugger.
- **JVM TI** (Java VM Tool Interface) API defines the debugging services a VM provides and a native programming interface for use by debugging, monitoring, and profiling tools. It provides a way to inspect the state of running JVM and to modify and control execution of applications running in the JVM. JVM TI provides an event-based support for bytecode instrumentation, the ability to alter the Java virtual machine bytecode instructions. JVM TI clients are called *agents*.

While JVM TI is a native programming interface requiring agents to be written in C / C++, `java.lang.instrument` is a higher-level API on top of JVM TI, providing a Java programming interface for bytecode instrumentation, which is one of the most important

enablers of Java debugging.

Enabling Debug Mode in IDE

To enable debugging, the JVM has to be started in debug mode with JPDA enabled. If you are using a Tomcat server adapter in Liferay Dev Studio, IDE takes care of enabling JPDA, and you only need to start the server in debug mode.



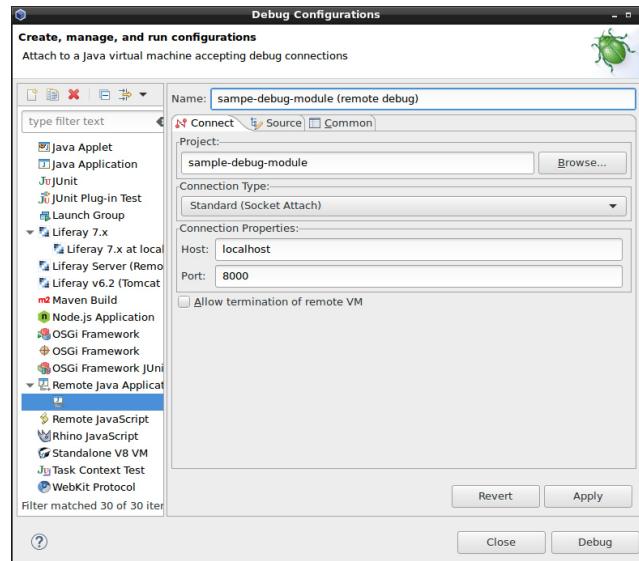
Tomcat can also be started manually in debug mode by simply adding the "jpda" startup option to catalina script:

```
TOMCAT_HOME/catalina.sh jpda start
```

Generally, servlet containers and Java EE servers can be started in debug mode by adding appropriate Xagentlib options for the JVM:

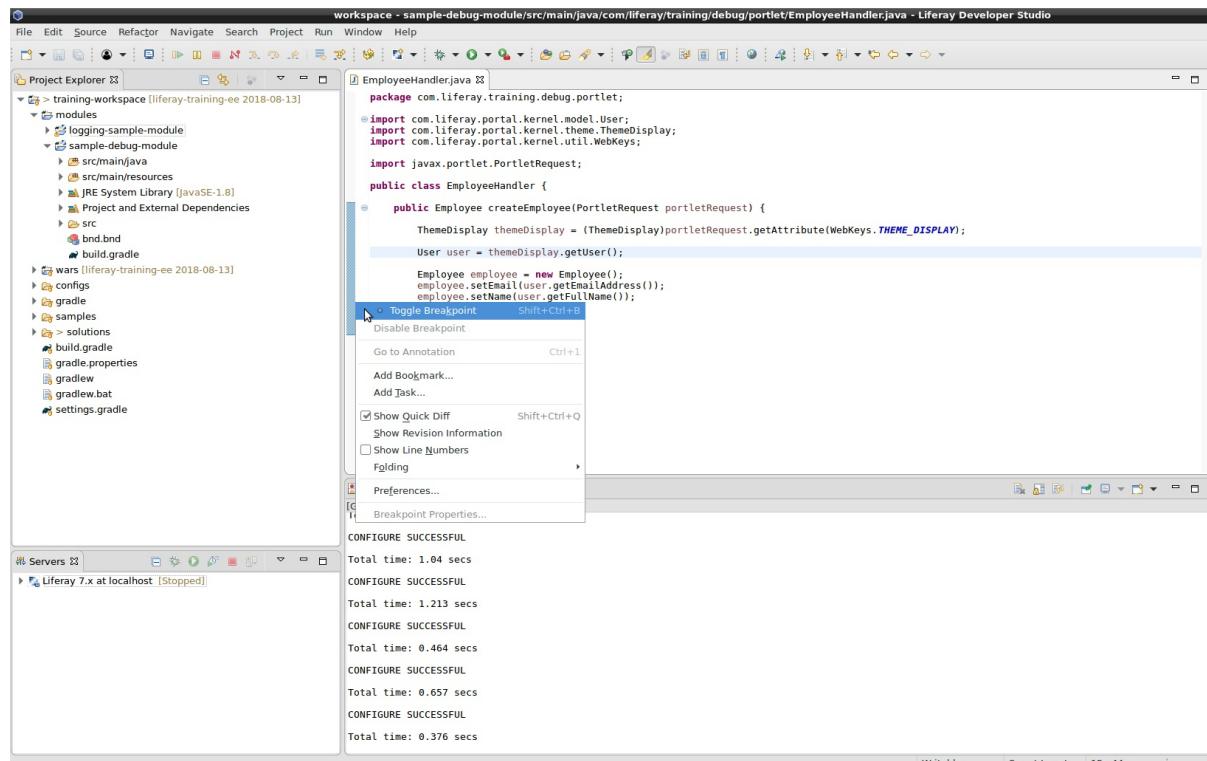
```
-Xagentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=8000
```

When debugging a remote server (including a server not launched from IDE), a remote debugging connection profile, corresponding to the JDWP options of JVM, has to be added in the IDE:

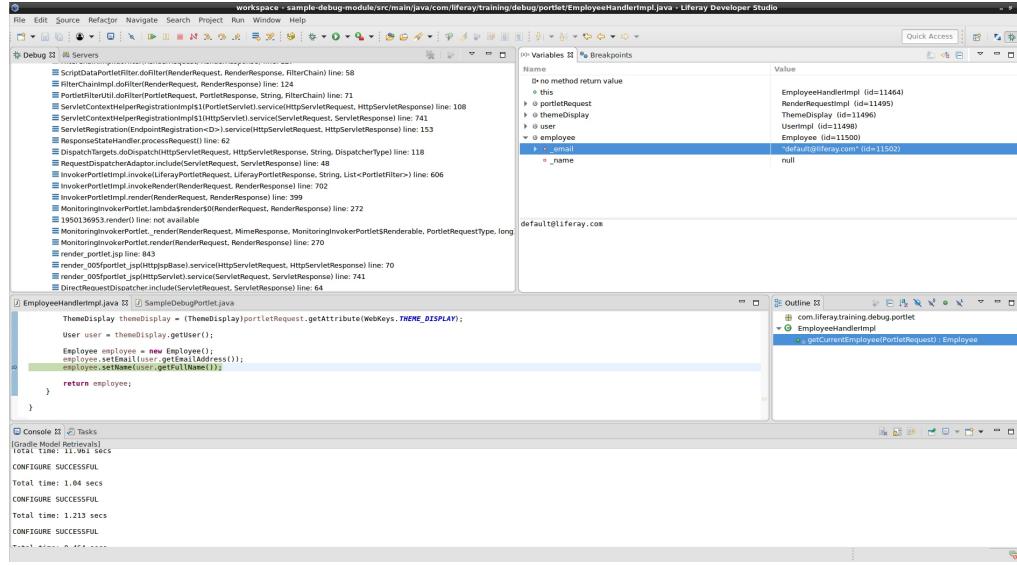


Debugging Basics

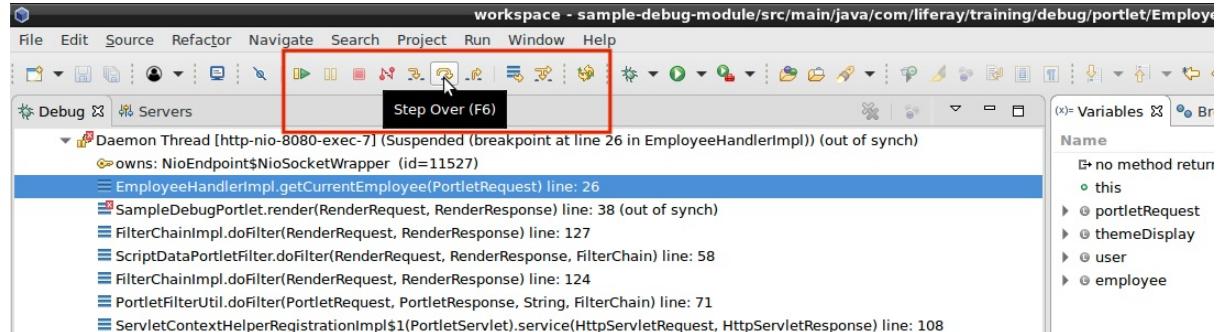
In running debugger, we are often interested to see why a certain variable gets an unprecedented value and what the application state is when some code block, method, or call is reached. For that purpose, we set *breakpoints*, which can be static, meaning that we always stop the execution at that point, or conditional, meaning that the execution only stops when a certain condition is met. We can also set variable watchpoints, which are triggered when a variable value is changed.



At the breakpoint, we can not only watch but also change variable values, for example, to check if the issues we were resolving are dependent on that specific value. Double-clicking the value allows you to change it.



When we have an idea about the area where the issue comes from, we can set a breakpoint and use debugger stepping functionalities to execute the code line by line forward, step into a method call, or even go backwards in the execution:



Setting Up Liferay Source Code for Debugging

Sometimes you might want to put a breakpoint in Liferay's code to see what happens at some specific point. For that purpose, you have to set up the portal source code for the IDE. Liferay provides source code packages for all the portal releases, including fix packs. To set up the source code, you need to:

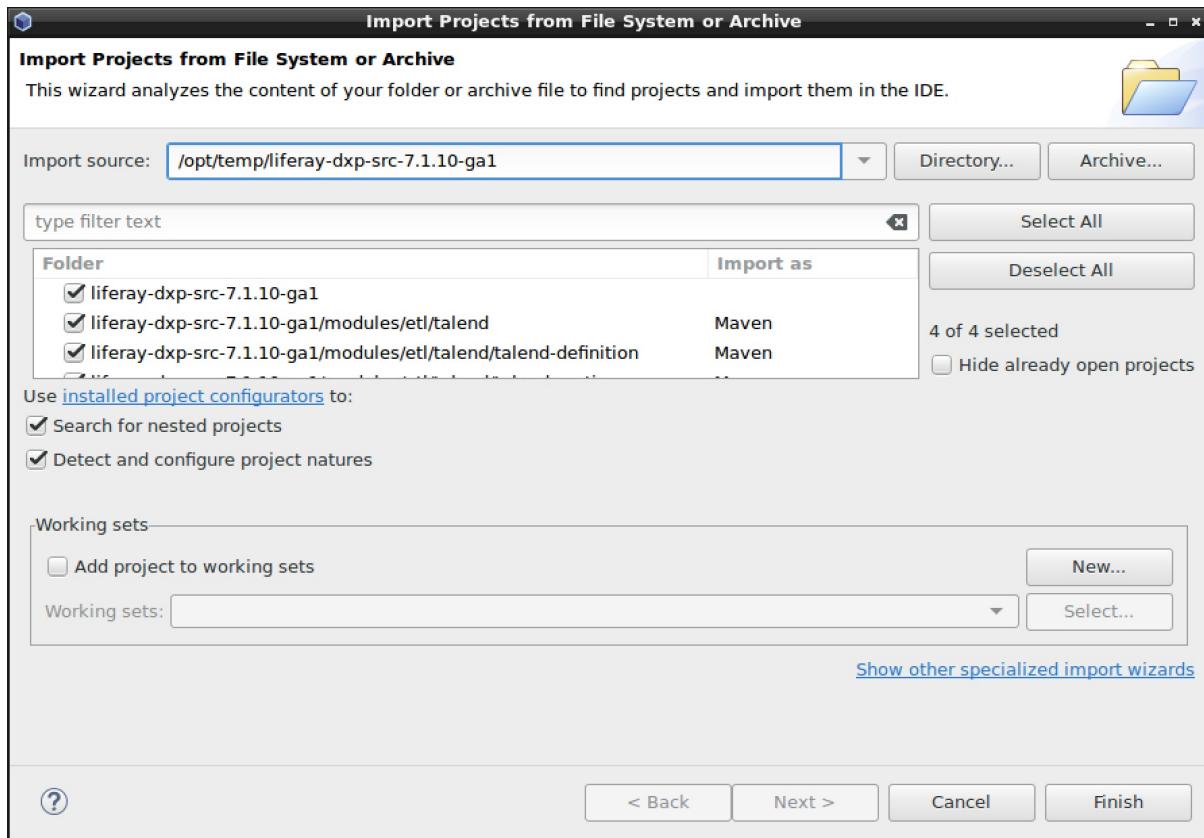
1. Download and extract the source code package
2. Import the source code project into a workspace
3. Set up the debugger configuration (if remote)

Step 1 - Download the Source Code

Liferay commercial release source codes can be downloaded from the customer portal at <https://customer.liferay.com/>.

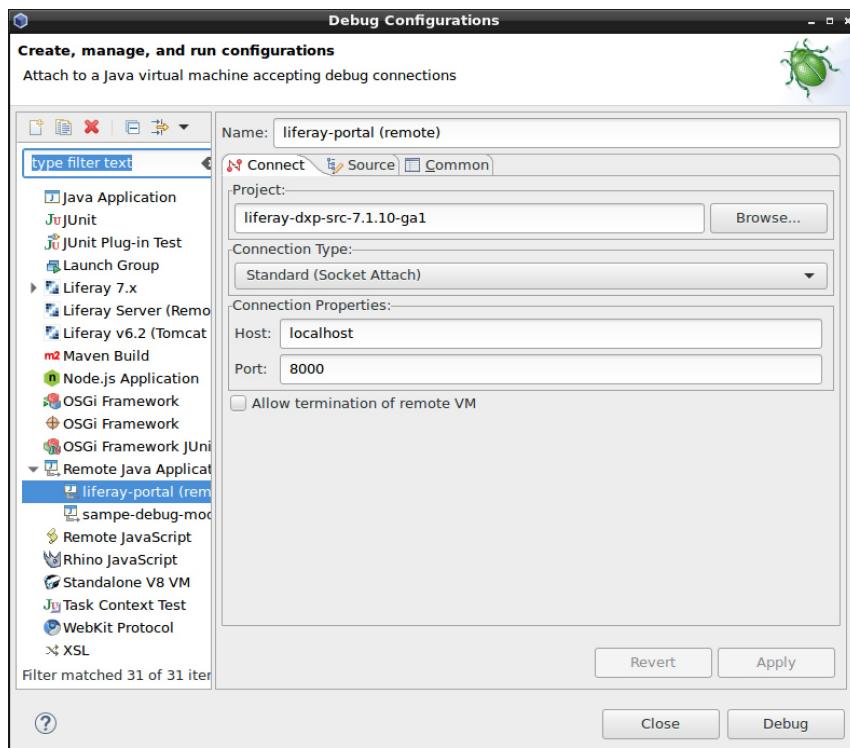
Step 2 - Import the Source Code into the Workspace (Eclipse)

When you extract the source code package and try to import that into Eclipse, you might notice there are no Eclipse project files present. To import the code into your workspace and auto-create the project files, use *File → Open Projects* from the File System:

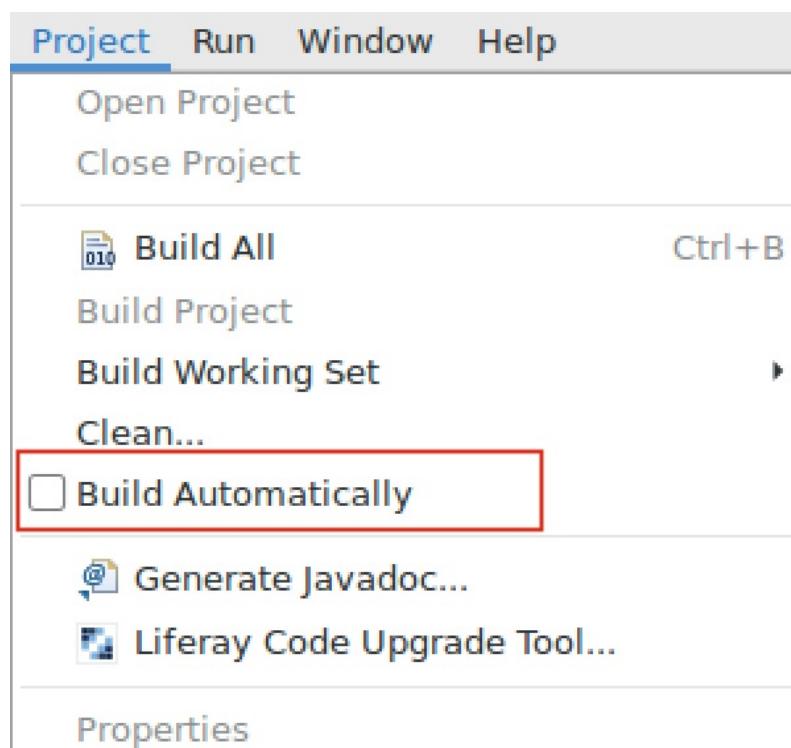


Step 3 - Set Up Debugger Configuration (Remote Debugging)

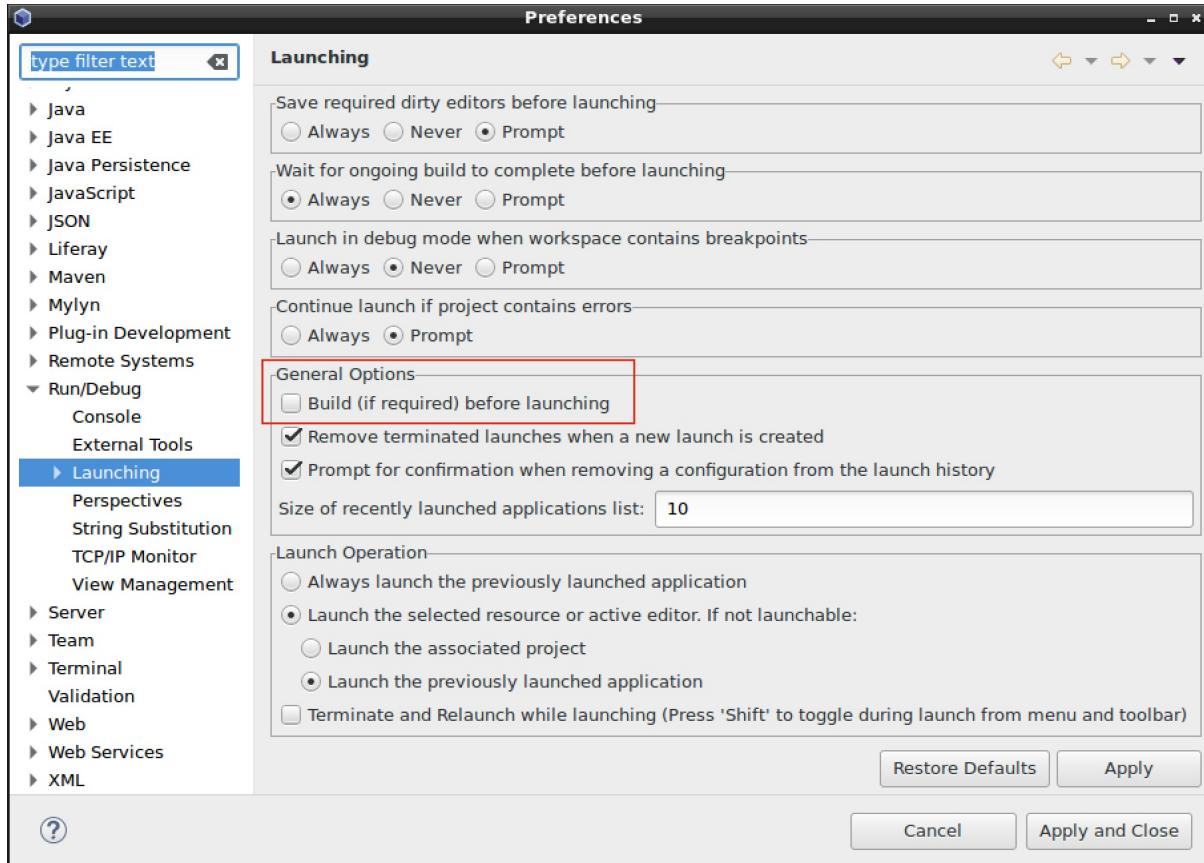
After the source code project is set up in the workspace, you need to create a debugger configuration for it:



Now you can launch the debugging session profile. Before launching, you should check one more thing. When you alter the portal source code or launch the debugging profile, the IDE tries to build the project automatically by default, which might fail or just take a lot of time. Disable automatic project-building in the project's settings:



Also disable building at launch in your Eclipse workspace preferences in *Window → Preferences → Run/Debug → Launching:*



Overview of Debugging in Production

Debugging in production usually has a different toolset. At the development stage, the development environment, source code, and IDE debugger are available, and often there's more time to resolve issues. The nature of debugging is forward-tracing, where you can run the program line by line to see what's causing or would possibly be causing an issue.

Testing also falls into this category of forward-tracing debugging.

Debugging issues in production systems is usually backward-tracing and relies on interpreting the issue symptoms from logs, stack traces, and thread dumps. Often, strict time constraints and security policies mean only a limited toolset is available.

Debugging in production often involves monitoring and profiling JVM and applications using, for example, APM tools. Here, we limit the scope of discussion to introducing thread dumps and JVM, getting memory statistics with Oracle JDK tools.

Thread Dumps

A thread dump shows information about what each thread is doing at a given time. This information includes the exact method call being executed and the thread state at that point in time.

Thread dumps are crucial in troubleshooting when the system is having performance issues and freezes. They show which threads and which function calls might be blocking the execution.

To create a thread dump of JVM with JDK tools, you first have to find the process ID of the running JVM. Depending on the operating system, there are multiple ways to get that information, but running the JDK tool `jps` from the *Command Line* is handy for this purpose:

```
jps -v
```

After you've found the process id (PID), you can use the JDK tool `Jstack` to create a thread dump of running JVM:

```
liferay@liferay> jstack -l PID > /opt/tmp/thread_dump.txt
```

The internal mechanics of thread dumps are beyond the scope of discussion here, but the excerpt below shows an example of a thread in the WAITING state:

```
...
"liferay/monitoring-1" #174 daemon prio=5 os_prio=0 tid=0x00007f45f1067800
nid=0x2be1 waiting on condition [0x00007f45a31fe000]
  java.lang.Thread.State: WAITING (parking)
    at sun.misc.Unsafe.park(Native Method)
      - parking to wait for  <0x00000000c7bcd0d0> (a java.util.concurren
t.locks.AbstractQueuedSynchronizer$ConditionObject)
        at java.util.concurrent.locks.LockSupport.park(LockSupport.ja
va:175)
        at java.util.concurrent.locks.AbstractQueuedSynchronizer$Cond
itionObject.await(AbstractQueuedSynchronizer.java:2039)
        at com.liferay.portal.kernel.concurrent.TaskQueue.take(TaskQu
```

```
eue.java:258)
    at com.liferay.portal.kernel.concurrent.ThreadPoolExecutor._g
etTask(ThreadPoolExecutor.java:548)
    at com.liferay.portal.kernel.concurrent.ThreadPoolExecutor.ac
cess$500(ThreadPoolExecutor.java:37)
    at com.liferay.portal.kernel.concurrent.ThreadPoolExecutor$Wo
rkerTask.run(ThreadPoolExecutor.java:672)
    at java.lang.Thread.run(Thread.java:748)
```

Locked ownable synchronizers:

- None

```
"LCS Worker 4" #171 prio=5 os_prio=0 tid=0x00007f45e002b800 nid=0
x2bde waiting on condition [0x00007f45ac16d000]
    java.lang.Thread.State: WAITING (parking)
    ...
    ...
```

If you had a stability issue in your system, you'd probably start looking for threads in the **BLOCKED** state, stealing the available threads and blocking the program execution, possibly caused by certain method calls. Then you would start to investigate why those calls are blocking the execution.

When debugging issues in production, it's good to remember the time aspect: a thread could, for example, be blocked at some point in time, and that would be completely normal. If that very same thread would be blocked after 30 seconds, you'd probably be having an issue. That's why, when taking thread dumps, it's important to take several thread dumps during the problematic application state.

Monitoring Memory with JStat

JVM problems are often memory-related. Your application might be leaking memory resources, or there might just be a resource allocation problem during a peak load. The JDK tool [JStat](#) shows statistics of JVM memory usage and garbage collection. It both helps in detecting the memory leaks and in tuning the JVM memory and garbage collection parameters.

The *Command Line* example below shows the garbage collection and memory statistics with the sample rate of 500ms:

liferay@liferay-VirtualBox \$ jstat -gcutil -t 27284 500ms								
Timestamp		S0	S1	E	O	M	CCS	YGC
YGCT	FGC	FGCT	GCT					
193093.6	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193094.1	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193094.6	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193095.1	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193095.6	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193096.1	24.19	0.00	95.19	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193096.6	24.19	0.00	97.09	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193097.1	24.19	0.00	98.98	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193097.6	24.19	0.00	98.98	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193098.1	24.19	0.00	98.98	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193098.6	24.19	0.00	100.00	57.11	91.63	82.49		450
10.111	8	3.835	13.947					
193099.1	0.00	95.58	1.89	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193099.6	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193100.1	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193100.6	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193101.1	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193101.7	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					
193102.2	0.00	95.58	1.99	57.14	91.63	82.49		451
10.125	8	3.835	13.960					

Knowledge Check

- Debugging covers a number of _____ under its umbrella.
- _____ is a collection of Java APIs used to facilitate debugging, profiling, and monitoring local and remote Java applications.
- Debugging issues in production systems is usually _____ and often relies on interpreting the symptoms from _____, _____, and _____.

Managing Deployment Issues

In this section, we discuss module deployment-related issues and some methods for resolving them. We've already learned that only the Liferay web application itself is deployed to and managed by the Java application server. All Liferay applications run in a Liferay-embedded OSGi container, and that's why resolving module deployment issues typically involves Gogo Shell.

Deploying Modules Overview

There are multiple ways to deploy and undeploy Liferay modules. Understanding these methods is essential in troubleshooting, as the changes they make in the file system depend on the method.

Autodeploy

Autodeploy is the default method of deploying modules to Liferay. You copy the module JAR or legacy WAR into the autodeploy folder, and it gets automatically deployed to the OSGi container.

By default, the autodeploy folder is `LIFERAY_HOME/deploy`, but it can be configured with `auto.deploy` prefixed settings in the [portal properties](#).

When a module JAR is read from the autodeploy folder, several things happen in the file system:

- JAR is copied to `LIFERAY_HOME/osgi/modules` .
- JSPs are compiled to `LIFERAY_HOME/work` .
- Runtime state is copied to `LIFERAY_HOME/osgi/state` .
- When using Tomcat, static resources are cached in:
 - `TOMCAT_HOME/work`
 - `TOMCAT_HOME/temp`

It's worth noticing that while the autodeploy method copies the module JARs to `LIFERAY_HOME/osgi/modules`, not all the deployment methods do that, and for the OSGi container runtime, it's not even needed: bytecode in `LIFERAY_HOME/osgi/state`

is from where the OSGi container uses the module, and you shouldn't ever edit contents of that folder runtime. If you delete the module JAR in the modules folder, the module will get undeployed.

Note that the autodeploy mechanism is not cluster-aware; modules deployed to a node's autodeploy folder are deployed only to that node.

Deploying from the Control Panel

In scenarios where autodeploy folder access is restricted, custom modules can also be deployed from the Control Panel. This method still uses the autodeploy folder internally, meaning that module JARS will also be copied to the `LIFERAY_HOME/osgi/modules` folder:

The screenshot shows the Liferay DXP Control Panel with the 'App Manager' section selected. The left sidebar shows 'Control Panel' with 'Apps' expanded, and 'App Manager' is highlighted. The main area is titled 'App Manager' with tabs for 'Apps' and 'Independent Modules'. A search bar is at the top. Below it, there's a table for 'APPS' with the following data:

Icon	Name	Version	Status	Actions
	Independent Modules	Version: 1.0.0	Status: Active	...
	Liferay Adaptive Media	Liferay Adaptive Media 1.0.0 requires the installation of the Liferay Digital Enterprise 7.0 Fix Pack 35 or higher. The Adaptive Media app tailors media in your portal to the device consuming it. Since users often consume media on multiple devices that have different screen sizes and capabilities, you should make sure that your portal presents that media in a manner suitable for each device. For... Suite: Liferay Collaboration Version: 2.0.0 Status: Active	Version: 2.0.0	...
	Liferay Alloy	Liferay Suite: Liferay Foundation Version: 1.0.0 Status: Active		...
	Liferay Analytics	Liferay Analytics Suite: Liferay Foundation Version: 1.0.0 Status: Active		...
	Liferay Announcements	Liferay ↳ Liferay Collaboration		...

Deploying from the Gogo Shell

Modules can be installed from within the Gogo Shell directly using the standard Felix Shell commands:

```
Trying 127.0.0.1...
Connected to localhost.
```

```
Escape character is '^]'.

```

```
Welcome to Apache Felix Gogo
```

```
g! install file:/opt/install/com.liferay.training.bundle-1.0.jar
Bundle ID: 961
```

A few things should be noticed with this method:

- This method doesn't start the bundle automatically.
- The module JAR is not copied to `LIFERAY_HOME/osgi/modules` .

Deploying in Dev Studio

When you deploy the modules in Liferay IDE using hot deploy, the autodeploy folder is not used. This method copies the compiled bytecode directly to
`LIFERAY_HOME/osgi/state/BUNDLE_ID` .

Deploying Using Blade

Modules can be deployed using the Blade CLI. This method detects the running JVM automatically, building and deploying the module directly to
`LIFERAY_HOME/osgi/state/BUNDLE_ID` . This method can only be used for JARS and not for legacy WARS:

```
liferay@liferay$ blade deploy
```

There are some advantages in using this method in your development environment. When run in the folder hierarchy, this command can deploy all the modules in the subdirectories too. The `blade watch` command is useful if you want to use the "hot deploy" feature directly from the *Command Line*. This command watches changes in the source folder and deploys changes automatically:

```
liferay@liferay$ blade deploy -w
```

Deploying Using the Gradle Wrapper Script

This method is meant to be used with Liferay Workspace with a Tomcat bundle. It tries to copy the compiled module JAR into `WORKSPACE_DIR/bundles/osgi/modules` :

```
liferay@liferay$ ./gradlew -w
```

Deploying Java EE Style WAR Modules

Liferay supports deploying legacy Java EE WAR-style web applications into the OSGi Container. When a WAR file is copied to the autodeploy folder, it gets automatically converted into a WAB bundle, which is then deployed to the OSGi Container. The WAB is an OSGi bundle and stands for *Web Application Bundle*. The WAB is specified in OSGi Compendium.

Liferay's WAB conversion mechanism creates an OSGi-compatible `MANIFEST.MF` file and a compatible folder structure. When a WAR is deployed, the following happens:

1. A WAR archive is copied to `LIFERAY_HOME/deploy` .
2. Liferay generates a WAB bundle on the fly.
3. Bundle runtime bytecode is created in `LIFERAY_HOME/osgi/state/BUNDLE_ID` .
4. The WAR archive gets stored in `LIFERAY_HOME/osgi/war` .

By default, the generated WABs are not stored but you can change the behavior in portal properties:

```
module.framework.web.generator.generated.wabs.store=true
module.framework.web.generator.generated.wabs.store.dir=${module.framework.base.dir}/wabs
```

Note on Marketplace Applications

Liferay Marketplace applications are packaged in Liferay's proprietary LPKG package format. Although the autodeployment process of LPKG packages is the same as for any deployable package type, the LPKG packages are stored in `LIFERAY_HOME/osgi/marketplace` .

Undeploying Modules Overview

Let's take a look at the common different undeployment methods and some caveats in using them.

Deleting a Module JAR

If the deployment method copied the JAR to

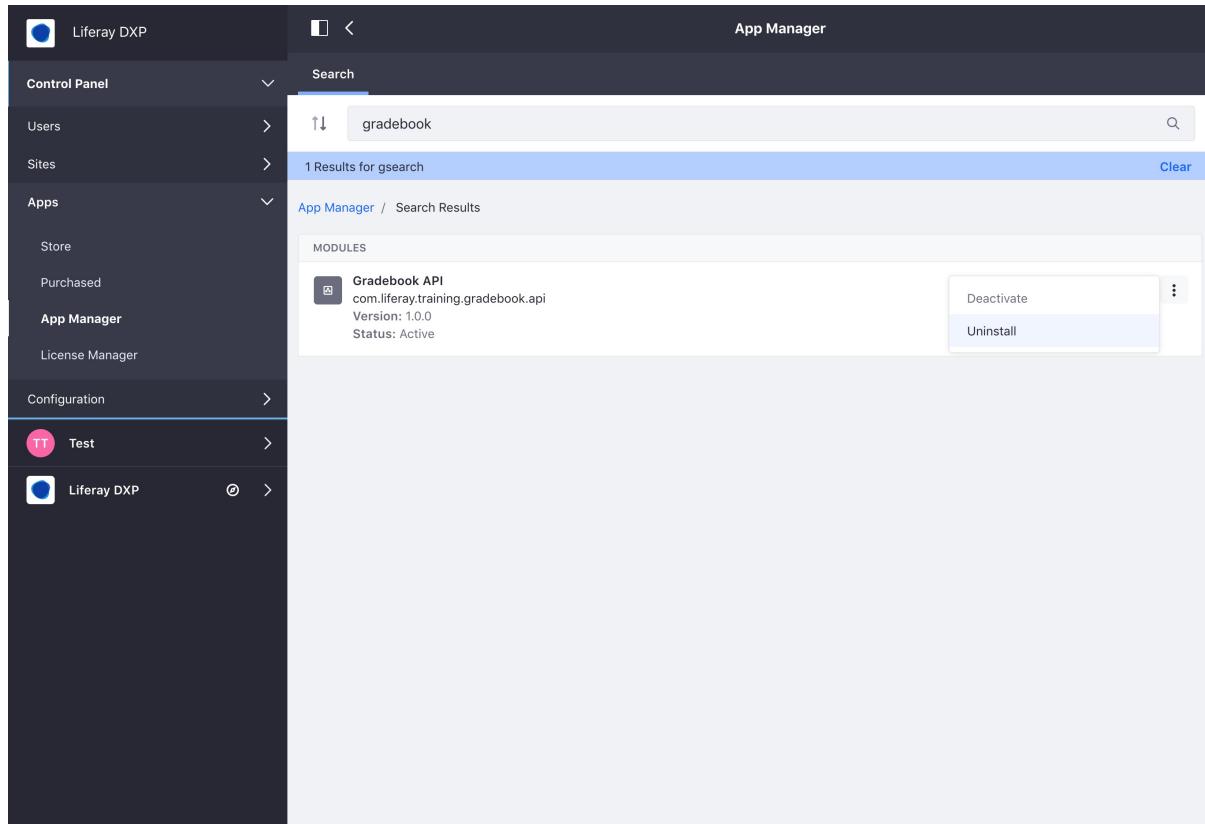
`LIFERAY_HOME/osgi/modules/MODULE.jar`, removing this file undeploys the module automatically. This is the most common use case.

Undeploying from the Control Panel

Undeploying the module from the Control Panel removes the module state from

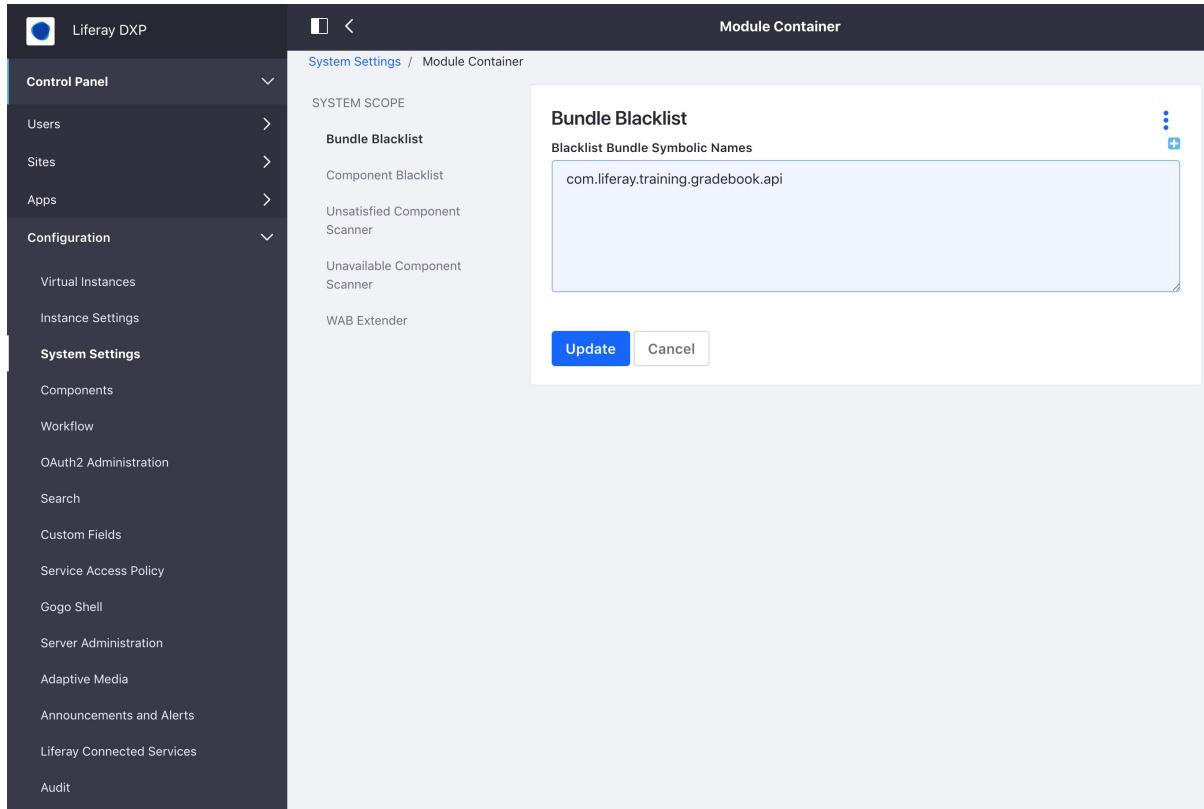
`LIFERAY_HOME/osgi/state`, but does not remove the module JAR from

`LIFERAY_HOME/osgi/modules`:



A couple things are worth noticing when this method is used. Undeploying from the Control Panel adds an entry for the Blacklister module, preventing a reinstall of the module. Before you can reinstall, you have to remove the blacklist entry. Another thing is that

removing the blacklisting automatically reinstalls the module if the JAR is still present in
LIFERAY_HOME/osgi/modules :



Undeploying from Gogo Shell

Modules can be uninstalled directly from the OSGi container using the `uninstall` command. This method does not remove the module JAR from

LIFERAY_HOME/osgi/modules :

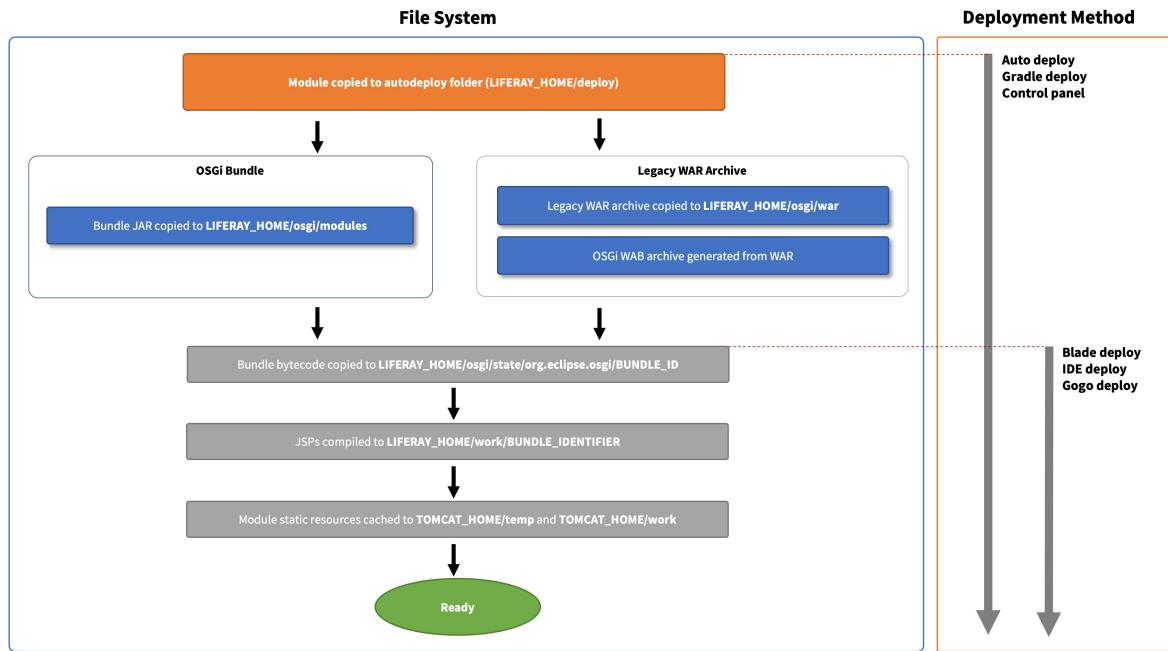
```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

```
Welcome to Apache Felix Gogo
```

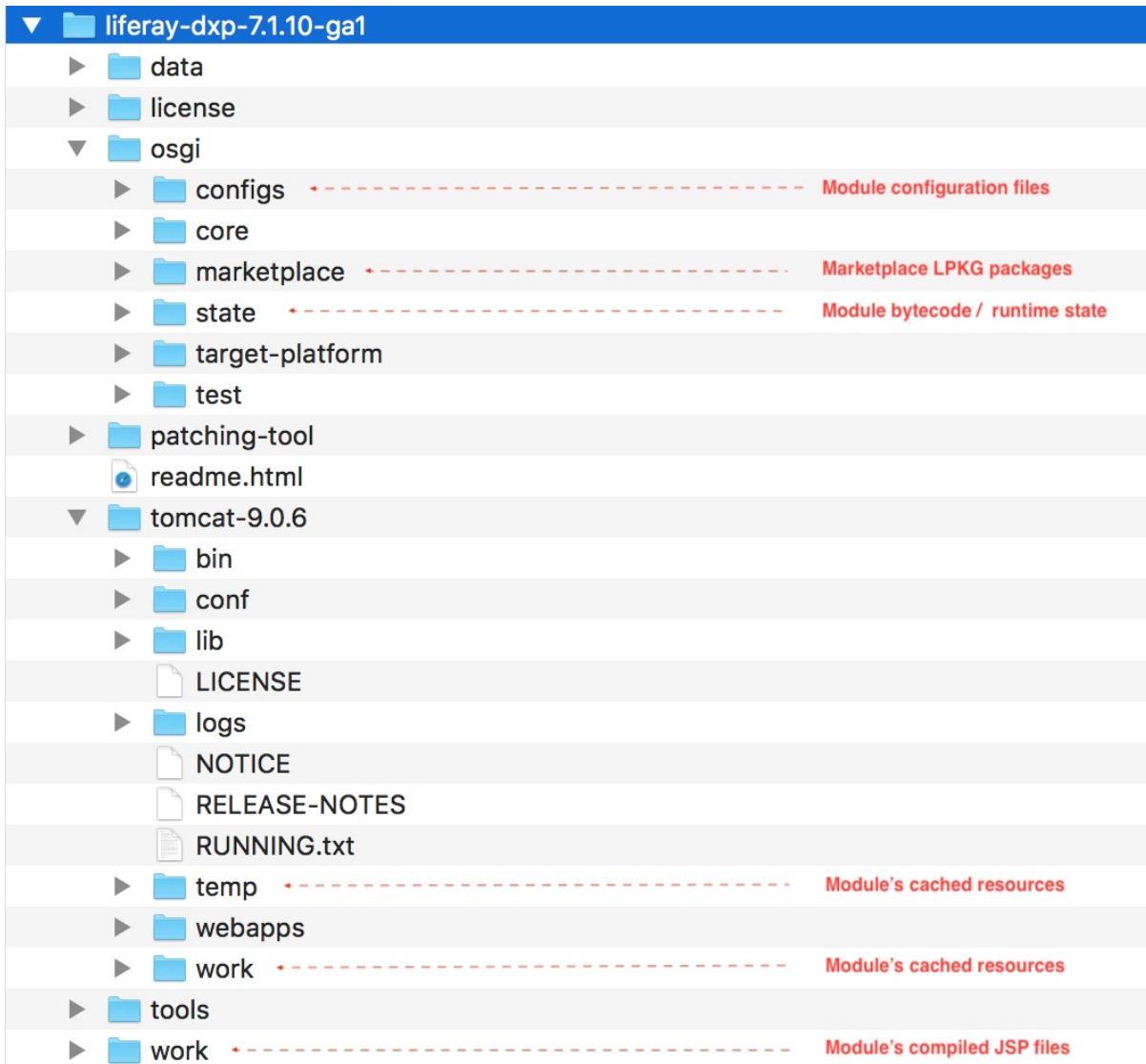
```
g! uninstall 961
```

Module Deployment Recap

Understanding the file-level changes during deployment helps to troubleshoot deployment-related issues. The following diagrams summarize the changes during the deployment process:



As seen in the folder structure:



Troubleshooting Deployment Issues

Steps to clean the platform and the module state completely are generally as follows:

1. Undeploy the module.
2. Shut down the portal.
3. Delete the module JAR, WAR, or LPKG.
4. Delete the module resources from `LIFERAY_HOME/osgi/state`.
5. Delete the module configuration file `LIFERAY_HOME/osgi/configs`.
6. Delete the compiled JSPs from `LIFERAY_HOME/work`.
7. Delete the content application server's temp folders. In Tomcat:
 - `TOMCAT_HOME/temp/`
 - `TOMCAT_HOME/work/`

Let's discuss some of the common module deployment-related scenarios.

Undeployed Module Reappears After Restart

Symptoms

The undeployed module is installed and active after restart.

Possible Cause

The module JAR, WAR, or LPKG file was not removed by the undeploy method.

Suggested Solution

Delete the module JAR, LPKG, or WAR file from the respective folder.

Module Redeployment Fails

Symptoms

Redeploying a module fails. The module is either not deployed at all or just doesn't start.

Possible Cause

An old module JAR file might be blocking the redeployment. This might happen, for example, if you were using Gradle deploy or autodeploy and **undeployed** the module from within Gogo Shell, which doesn't remove the module JAR. There might also be a blacklist entry for the module.

Possible Solutions

- Delete the module JAR from `LIFERAY_HOME/osgi/modules`.
- Check blacklisting settings in *Control Panel → Configuration → System Settings → Platform → Bundle Blacklist*.

Module Resources Not Refreshing

Symptoms

Redeploying the module won't refresh static resources like CSS and JavaScript files or compiled JSP files.

Possible Cause

Caching settings might prevent resources from refreshing. There might also be a system time skew or faulty timezone settings letting the system know that current resources are newer than the updated ones.

Possible Solutions

- Check that the portal is running on development mode.
- Check that both the development and server machine clocks are synchronized.
- Check that the portal JVM is running on either the GMT or UTC timezone.

To clear the JSP cache in production, you can also try to clear the direct servlet cache from *Control Panel → Server Administration → Resources → Clear the direct servlet cache*.

Module Gets Installed But Doesn't Start

This issue type usually falls into two possible categories:

1. Unsatisfied module dependencies
2. Unsatisfied module requirements

Unsatisfied Module Dependencies

Symptoms

The module deploys but remains in the *installed* state and doesn't become *active*.

Possible Cause

The module didn't make it to the *resolved* state because some of the module's dependencies are missing or couldn't get activated. This might also happen because of the wrong dependency compile scope.

Possible Solutions

- Check the compile scopes in `build.gradle` if any resource should be included in

the build (`compileInclude`).

- Use the Gogo `diag` command to find any missing dependencies in the container.

Unsatisfied Module Requirements

Symptoms

The module is in the *active* state, but some of its service components seem not to be working.

Possible Cause

OSGi bundles and components have different lifecycles; a bundle may start even if some of its components do not. Usually this happens when some of the component's references (`@Reference`) are not satisfied, i.e., not found or started. There might also be a circular reference.

Possible Solutions

- Use the Gogo Shell to find the failing component id by running `ds:unsatisfied BUNDLE_ID`. Then run `scr:info COMPONENT_ID` to find unsatisfied references. Find out why the reference was not available.
- Run `ds:softCircularDependency` to detect circular references.

Optional Exercises

Tasks to Accomplish

- Use the remote service API for testing the service
- Extend the Gradebook to support student submissions for Assignments
- We'll demonstrate how to use the Liferay NPM bundler to add an NPM managed Javascript resource to our project
- Create a Gradebook application configuration in the API module and consume that from both the service and web modules
- Enable workflow support for the Gradebook assignments
- Add a REST API to the service layer by way of a JAX-RS Whiteboard
- Add logging to the Gradebook service, so that every time an assignment is created, a log entry is created
- Create a new test module and create an integration tests using the Liferay Arquillian JUnit bridge
- Learn debugging basics:
 - How to start the Tomcat server in debug mode
 - How to set breakpoints
 - how to control program execution with debugger steps
- Practice using Liferay Log and Gogo shell for troubleshooting failing module deployment

Optional Exercise

Test the Service Through Remote API

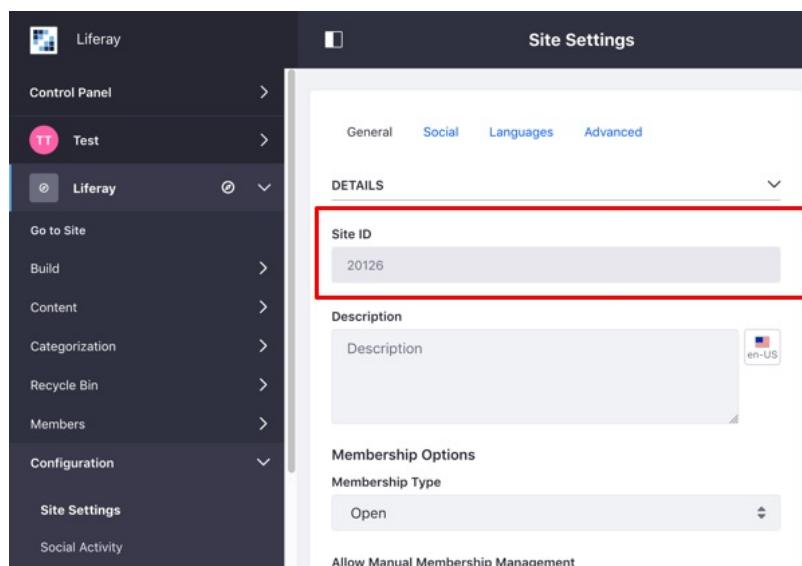
Exercise Goals

- Find the test site ID
- Add an Assignment using browser console
- Get an Assignment through JSON API test page
- Delete an Assignment through JSON API test page

Find the Test Site ID

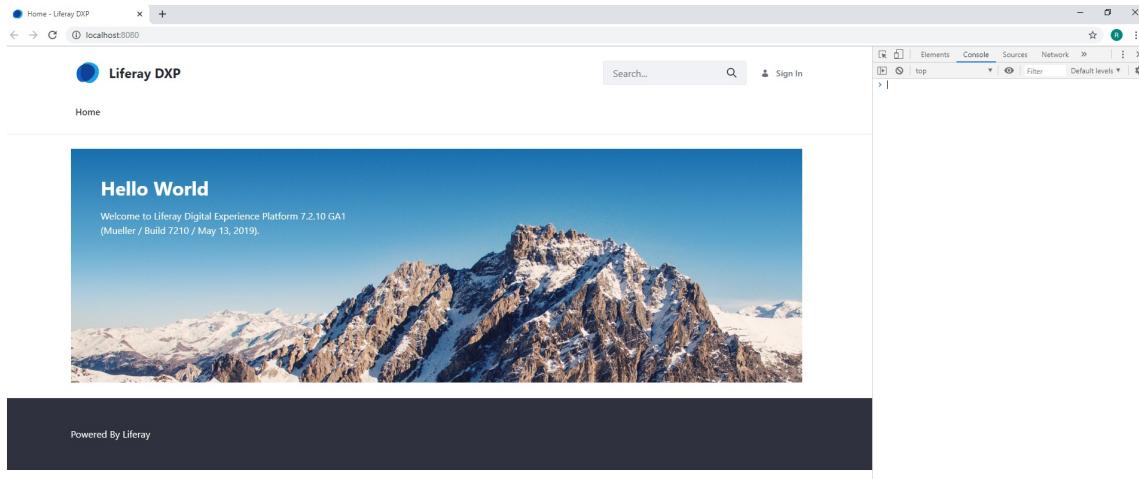
We have to find an ID (`groupId`) for the site where we are going to create our test assignments. We'll be using the default guest site for this exercise:

1. **Sign in** to Liferay with your web browser.
2. **Open** the site administration panel in the *Product Menu*.
3. **Click on Configuration → Settings**.
4. **Find the Site ID value.**



Add an Assignment Using Browser Console

1. **Make sure** that you are logged in to the portal.
2. **Open** the JavaScript console of your favorite browser (Usually **Ctrl+Shift+J** (Windows / Linux) or **Cmd+Opt+J** (OSX))



3. **Use** following JSON snippet in the Javascript console (**check that the site ID matches**):

```
Liferay.Service('/gradebook.assignment/add-assignment', {  
    groupId: 20123,  
    titleMap: { 'en_US': 'How to make a birthday cake' } ,  
    descriptionMap: { 'en_US': 'Design most delicious and bea  
utiful birthday cake.'},  
    dueDate: (new Date('2019-08-22')).getTime()  
}, function(obj) {  
    console.log(obj);  
});
```

- You should get an assignment as JSON as a response:

```

> Liferay.Service('/gradebook.assignment/add-assignment', {
    groupId: 20126,
    title: { 'en_US': 'How to make birthday cake' },
    description: 'Design most delicious and beautiful birthday cake.',
    dueDate: (new Date('2018-05-22')).getTime()
}, function(obj) {
    console.log(obj);
});
< ▶ {readyState: 1, getResponseHeader: f, getAllResponseHeaders: f, setReque
e: 1526947200000, ...} ⓘ
    assignmentId: "101"
    companyId: "20099"
    createDate: 1529966843319
    description: "Design most delicious and beautiful birthday cake."
    dueDate: 1526947200000
    groupId: "20126"
    modifiedDate: 1529966843319
    status: 0
    statusByUserId: "0"
    statusByUserName: ""
    statusDate: null
    title: "<?xml version='1.0' encoding='UTF-8'?><root available-locales=
    titleCurrentValue: "How to make birthday cake"
    userId: "20139"
    userName: "test"
    uuid: "40e0b07d-5ff4-dbbe-092d-b4b72e0caeba"
    ► __proto__: Object

```

- Find the `assignmentId` in the JSON response, that is required at the next step.

Get an Assignment through JSON API Test Page

- Open <http://localhost:8080/api/jsonws> in your web browser.
- Choose *Gradebook* in the *Context Name* menu.

The screenshot shows a web interface for testing JSON Web Services. At the top, it says "JSONWS API". Below that, there's a dropdown menu labeled "Context Name" with "gradebook" selected. To the right of the dropdown is a message: "Please select a method on the left." Below the dropdown is a "Search" input field. On the left side, there's a sidebar with a tree view. Under the "Assignment" category, the "add-assignment" method is highlighted in blue, indicating it's the selected method for testing.

On the menu, you'll see a list of methods we just added to our remote service. We'll now test our service with a browser's Javascript console.

1. **Click** *get-assignment* on the JSONWS API test page menu.
2. **Enter** the assignment ID.
3. **Click** *Invoke*.

Delete an Assignment through JSON API Test Page

1. **Click** *delete-assignment* on the page menu.
2. **Enter** the assignment ID.
3. **Click** *Invoke*.

Optional Exercise

Implement Submissions

Exercise Goals

- Implement the service layer:
 - Define the Submission entity in `service.xml`
 - Increase submission text column size by modifying the `portlet-model-hints.xml`
 - Implement `SubmissionLocalServiceImpl`
 - Implement `SubmissionServiceImpl`
- Implement the controller layer
 - Implement the MVC Commands
- Implement the user interface
 - Implement the JSP files
- Test the application

Notice that we have already created some resources for this exercise in the previous exercises:

- Permission definitions in the `default.xml` of *gradebook-service* module.
- Localizations in the `Language.properties` of the *gradebook-web* module
- MVC Command name constants in the `MVCCCommandNames.java` class in the *gradebook-web* module.

Define the Submission entity in `service.xml`

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

2. **Implement** the submission entity definition right after the closing `<entity>` tag of Assignment:

```

<entity name="Submission" local-service="true" remote-service="true">

    <column name="submissionId" type="long" primary="true" />

    <!-- Group instance -->

    <column name="groupId" type="long" />

    <!-- Audit fields -->

    <column name="companyId" type="long" />
    <column name="userId" type="long" />
    <column name="userName" type="String" />
    <column name="createDate" type="Date" />
    <column name="modifiedDate" type="Date"/>

    <column name="assignmentId" type="long"/>
    <column name="studentId" type="long"/>
    <column name="submitDate" type="Date"/>
    <column name="submissionText" type="String" />
    <column name="comment" type="String"/>
    <column name="grade" type="int"/>

    <finder name="GroupId" return-type="Collection">
        <finder-column name="groupId"></finder-column>
    </finder>
    <finder name="G_A" return-type="Collection">
        <finder-column name="groupId"></finder-column>
        <finder-column name="assignmentId"></finder-column>
    </finder>
    <finder name="StudentId" return-type="Collection">
        <finder-column name="studentId"></finder-column>
    </finder>
    <finder name="StudentIdAssignmentId" return-type="Collection">
        <finder-column name="studentId"></finder-column>

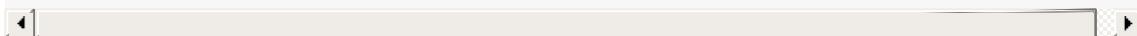
```

```

<finder-column name="assignmentId"></finder-column>
</finder>

<!-- References -->

<reference package-path="com.liferay.portlet.asset" entity
="AssetEntry" />
<reference package-path="com.liferay.portlet.asset" entity
="AssetTag" />
</entity>
```



The reference field between Assignment and Submission is
assignmentId .

3. Add `SubmissionValidationException` definition in the `<exceptions>` tag:

```

<exceptions>
  <exception>AssignmentValidation</exception>
  <exception>SubmissionValidation</exception>
</exceptions>
```

4. Rebuild the service to generate the Submission services.

Increase Submission Text Column Size by Modifying the `portlet-model-hints.xml`

The default size for a text column is 75 characters. Increase the size to 1024 characters:

1. Open the file `src/main/resources/META-INF/portlet-model-hints.xml` in the *gradebook-service* module.
2. Find the definition of `submissionText` column and replace it with:

```

<field name="submissionText" type="String">
  <hint name="max-length">1024</hint>
</field>
```

3. Rebuild the service.

Implement Message Stacking in the Submission Validation Exception

When validating entities, it's often desired to get all validation errors on the screen at once. Let's customize the submission validation exception and add a new constructor taking a list as a parameter:

1. **Open** the class

```
com.liferay.training.gradebook.exceptionSubmissionValidationException
```

in the *gradebook-api* module.

2. **Implement** the new constructor, getter method and list variable to the class as follows:

```
/*
 * Custom constructor for validation
 * @param errors
 */
public SubmissionValidationException(List<String> errors) {
    super(String.join(", ", errors));
    _errors = errors;
}

public List<String> getErrors() {
    return _errors;
}

private List<String> _errors;
```

3. **Organize** missing imports.

Implement SubmissionLocalServiceImpl

Implement the CRUD logic for the Submission entities:

1. **Open** the class

```
com.liferay.training.gradebook.service.implSubmissionLocalService
```

ceImpl in the *gradebook-service* module.

2. **Replace** the contents of the class with the following:

```
/*
 * 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 publ  
 ished 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 us  
 eful, but WITHOUT  
 * ANY WARRANTY; without even the implied warranty of MERCCHAN  
 TABILITY or FITNESS  
 * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Publi  
 c 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.model.User;  
import com.liferay.portal.kernel.module.configuration.ConfigurationException;  
import com.liferay.portal.kernel.service.ServiceContext;  
import com.liferay.training.gradebook.exception.SubmissionVal  
idationException;  
import com.liferay.training.gradebook.model.Assignment;  
import com.liferay.training.gradebook.modelSubmission;  
import com.liferay.training.gradebook.service.base.Submission  
LocalServiceBaseImpl;  
  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
import org.osgi.service.component.annotations.Component;
```

```
/**  
 * The implementation of the submission local service.  
 *  
 * <p>  
 * All custom service methods should be put in this class. Whenever methods are added, rerun ServiceBuilder to copy their definitions into the <code>com.liferay.training.gradebook.service.SubmissionLocalService</code> interface.  
 *  
 * <p>  
 * This is a local service. Methods of this service will not have security checks based on the propagated JAAS credentials because this service can only be accessed from within the same VM.  
 * </p>  
 *  
 * @author Brian Wing Shun Chan  
 * @see SubmissionLocalServiceBaseImpl  
 */  
  
@Component(  
    property = "model.class.name=com.liferay.training.gradebook.model.Submission",  
    service = AopService.class  
)  
public class SubmissionLocalServiceImpl extends SubmissionLocalServiceBaseImpl {  
  
    /*  
     * NOTE FOR DEVELOPERS:  
     *  
     * Never reference this class directly. Use <code>com.liferay.training.gradebook.service.SubmissionLocalService</code> via injection or a <code>org.osgi.util.tracker.ServiceTracker</code> or use <code>com.liferay.training.gradebook.service.SubmissionLocalServiceUtil</code>.   
     */  
  
    /**  
     * Adds a new submissions  
     */
```

```
* @param assignmentId
* @param studentId
* @param submissionText
* @param serviceContext
* @return
* @throws PortalException
*/
@Override
public Submission addSubmission(
    long assignmentId, long studentId, String submissionText,
    ServiceContext serviceContext)
    throws PortalException {

    Assignment assignment =
        assignmentPersistence.findByPrimaryKey(assignmentId);

    long userId = serviceContext.getUserId();

    // Verify that user exists (throws exception if not).

    User user = userLocalService.getUser(userId);

    // Verify that student exists (throws exception if not).

    User studentUser = userLocalService.getUser(studentId);

    // Validate submission

    validateSubmission(
        serviceContext.getCompanyId(), studentId, assignment,
        submissionText);

    // Create submission id.

    long submissionId =
```

```
        counterLocalService.increment(Submission.class.get
tName());  
  
        // Create new submission.  
  
        Submission submission =
            submissionLocalService.createSubmission(submissio
nId);  
  
        submission.setSubmissionId(submissionId);
        submission.setAssignmentId(assignmentId);
        submission.setCompanyId(assignment.getCompanyId());
        submission.setGroupId(assignment.getGroupId());
        submission.setCreateDate(new Date());
        submission.setModifiedDate(new Date());  
  
        submission.setUserId(userId);
        submission.setGrade(-1);
        submission.setStudentId(studentId);
        submission.setSubmissionText(submissionText);
        submission.setSubmitDate(new Date());  
  
        return super.addSubmission(submission);
    }  
  
    @Override
    public Submission updateSubmission(
        long submissionId, String submissionText, ServiceCont
ext serviceContext)
        throws PortalException {  
  
        Submission submission = getSubmission(submissionId);  
  
        Assignment assignment =
            assignmentPersistence.findByPrimaryKey(submis
sion.getAssignmentId());  
  
        validateSubmission(
            serviceContext.getCompanyId(), submission.getStud
entId(),
```

```
        assignment, submissionId, submissionText);

    submission.setSubmissionText(submissionText);
    submission.setSubmitDate(new Date());

    return super.updateSubmission(submission);
}

public List<Submission> getSubmissionsByAssignment(
    long groupId, long assignmentId) {

    return submissionPersistence.findByG_A(groupId, assignmentId);
}

public List<Submission> getSubmissionsByAssignment(
    long groupId, long assignmentId, int start, int end)
{

    return submissionPersistence.findByG_A(
        groupId, assignmentId, start, end);
}

public int getSubmissionsCountByAssignment(
    long groupId, long assignmentId) {

    return submissionPersistence.countByG_A(groupId, assignmentId);
}

public Submission gradeSubmission(long submissionId, int grade)
throws PortalException {

    Submission submission = getSubmission(submissionId);

    submission.setGrade(grade);
    submission.setModifiedDate(new Date());

    return super.updateSubmission(submission);
```

```
}

public Submission gradeAndCommentSubmission(
    long submissionId, int grade, String comment)
throws PortalException {

    Submission submission = getSubmission(submissionId);

    submission.setGrade(grade);
    submission.setComment(comment);

    submission.setModifiedDate(new Date());

    return super.updateSubmission(submission);
}

private void validateSubmission(
    long companyId, long studentId, Assignment assignment
    ,
    long submissionId, String submissionText)
    throws PortalException, SubmissionValidationException
, ConfigurationException {

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

    // Validate the due date.

    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
    String dueDate = sdf.format(assignment.getDueDate());
    String today = sdf.format(new Date());

    if (dueDate.compareTo(today) < 0) {
        errorMessages.add("submissionTooLate");
    }

    // Validate submission count.

    if (submissionId < 0) {
```

```

        long submissionCount =
            submissionPersistence.countByStudentIdAssignmentId(
                studentId, assignment.getAssignmentId()));

        if (submissionCount > 0) {
            errorMessages.add("onlyOneSubmissionAllowed")
        }
    }
    else {

        Submission submission =
            submissionPersistence.fetchByPrimaryKey(submissionId);

        if (submission.getStudentId() != studentId) {
            errorMessages.add("onlyOneSubmissionAllowed")
        }
    }

    // Validate text length.

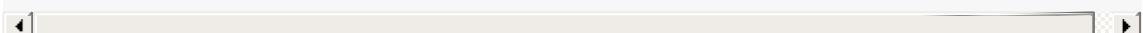
    if (submissionText == null) {

        errorMessages.add("submissionTextNull");
    }

    // Throw an exception if necessary.

    if (errorMessages.size() > 0) {
        throw new SubmissionValidationException(errorMessages);
    }
}
}

```



3. Rebuild the service.

Implement SubmissionServiceImpl

Implement the facade methods in the remote service implementation class:

1. **Open** the class

com.liferay.training.gradebook.service.impl.SubmissionServiceImpl
in the *gradebook-service* module.

2. **Implement** as follows:

```
/*
 * 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.auth.PrincipalException;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermission;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.training.gradebook.model.Assignment;
```

```
import com.liferay.training.gradebook.model.Submission;
import com.liferay.training.gradebook.service.base.Submission
ServiceBaseImpl;

import java.util.List;

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.ReferencePolicy
Option;

/**
 * The implementation of the submission remote service.
 *
 * <p>
 * All custom service methods should be put in this class. Whenever methods are added, rerun ServiceBuilder to copy their definitions into the <code>com.liferay.training.gradebook.service.SubmissionService</code> interface.
 *
 * <p>
 * This is a remote service. Methods of this service are expected to have security checks based on the propagated JAAS credentials because this service can be accessed remotely.
 * </p>
 *
 * @author Brian Wing Shun Chan
 * @see SubmissionServiceBaseImpl
 */

@Component(
    property = {
        "json.web.service.context.name=gradebook",
        "json.web.service.context.path=Submission"
    },
    service = AopService.class
)
public class SubmissionServiceImpl extends SubmissionServiceB
aseImpl {
```

```
/*
 * NOTE FOR DEVELOPERS:
 *
 * Never reference this class directly. Always use <code>
com.liferay.training.gradebook.service.SubmissionServiceUtil</
code> to access the submission remote service.
 */

public Submission addSubmission(
    long assignmentId, long studentId, String submissionText,
    ServiceContext serviceContext)
throws PortalException {

    // Check permissions.

    _assignmentModelResourcePermission.check(
        getPermissionChecker(), assignmentId, "ADD_SUBMISSION");

    return submissionLocalService.addSubmission(
        assignmentId, studentId, submissionText, serviceContext);
}

public Submission deleteSubmission(long submissionId)
throws PortalException {

    // Check permissions.

    Submission submission =
        submissionLocalService.getSubmission(submissionId);

    _assignmentModelResourcePermission.check(
        getPermissionChecker(), submission.getAssignmentId(),
        "DELETE_SUBMISSION");
```

```
        return submissionLocalService.deleteSubmission(submissionId);
    }

    public Submission gradeSubmission(long submissionId, int grade)
        throws PortalException {

        Submission submission =
            submissionLocalService.getSubmission(submissionId);

        _assignmentModelResourcePermission.check(
            getPermissionChecker(), submission.getAssignmentId(),
            "GRADE_SUBMISSION");

        return submissionLocalService.gradeSubmission(submissionId, grade);
    }

    public Submission gradeAndCommentSubmission(
        long submissionId, int grade, String comment)
        throws PortalException {

        Submission submission =
            submissionLocalService.getSubmission(submissionId);

        _assignmentModelResourcePermission.check(
            getPermissionChecker(), submission.getAssignmentId(),
            "GRADE_SUBMISSION");

        return submissionLocalService.gradeAndCommentSubmission(
            submissionId, grade, comment);
    }

    public List<Submission> getSubmissionsByAssignment(
```

```
long groupId, long assignmentId, int start, int end)
throws PrincipalException, PortalException {

    _assignmentModelResourcePermission.check(
        getPermissionChecker(), assignmentId, "VIEW_SUBMISSIONS");

    return submissionPersistence.findByG_A(
        groupId, assignmentId, start, end);
}

public int getSubmissionsCountByAssignment(
    long groupId, long assignmentId) {

    return submissionPersistence.countByG_A(groupId, assignmentId);
}

public Submission updateSubmission(
    long submissionId, String submissionText, ServiceContext serviceContext)
throws PortalException {

    Submission submission =
        submissionLocalService.getSubmission(submissionId);

    _assignmentModelResourcePermission.check(
        getPermissionChecker(), submission.getAssignmentId(),
        "EDIT_SUBMISSION");

    return submissionLocalService.updateSubmission(
        submissionId, submissionText, serviceContext);
}

@Reference(
    policy = ReferencePolicy.DYNAMIC,
    policyOption = ReferencePolicyOption.GREEDY,
    target = "(model.class.name=com.liferay.training.grad
```

```

    ebook.model.Assignment)"
)
private volatile ModelResourcePermission<Assignment>
    _assignmentModelResourcePermission;

}

```

3. **Rebuild** the service.

Implement the MVC Action Commands

1. **Add** and implement the three MVC Action Commands in the *gradebook-web* module for adding, editing, grading and deleting submissions:

- `com.liferay.training.gradebook.web.portlet.action.EditSubmissionMVCActionCommand`
- `com.liferay.training.gradebook.web.portlet.action.GradeSubmissionMVCActionCommand`
- `com.liferay.training.gradebook.web.portlet.action.DeleteSubmissionMVCActionCommand`

EditSubmissionMVCActionCommand

```

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

```

```
import com.liferay.training.gradebook.exception.SubmissionValidationException;
import com.liferay.training.gradebook.model.Submission;
import com.liferay.training.gradebook.service.SubmissionService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

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

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

/**
 * MVC Command for adding and editing submissions.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
        "mvc.command.name=" + MVCCommandNames.ADD_SUBMISSION,
        "mvc.command.name=" + MVCCommandNames.EDIT_SUBMISSION
    },
    service = MVCActionCommand.class
)
public class EditSubmissionMVCActionCommand extends BaseMVCActionCommand {

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

        ThemeDisplay themeDisplay =
```

```
(ThemeDisplay) actionRequest.getAttribute(WebKeys.THE  
ME_DISPLAY);  
  
try {  
  
    // Create service context.  
  
    ServiceContext serviceContext = ServiceContextFactory  
.getInstance(  
        Submission.class.getName(), actionRequest);  
  
    String submissionText =  
        ParamUtil.getString(actionRequest, "submissionTex  
t");  
  
    // Do edit or add action.  
  
    if (isEditAction(actionRequest)) {  
  
        long submissionId =  
            ParamUtil.getLong(actionRequest, "submissio  
nId");  
  
        // Call the service to update the submission.  
  
        _submissionService.updateSubmission(  
            submissionId, submissionText, serviceContext)  
;  
  
        // Set the success message  
  
        SessionMessages.add(actionRequest, "submissio  
nUp  
dated");  
  
    }  
    else {  
        long assignmentId =  
            ParamUtil.getLong(actionRequest, "assignmentI  
d");  
    }  
}  
}
```

```
        long studentUserId = themeDisplay.getUserId();

        _submissionService.addSubmission(
            assignmentId, studentUserId, submissionText,
            serviceContext);

        SessionMessages.add(actionRequest, "submissionAdd
ed");
    }

    sendRedirect(actionRequest, actionResponse);
}
catch (SubmissionValidationException e) {

    e.getErrors().forEach(key -> SessionErrors.add(action
Request, key));

    actionResponse.setRenderParameter(
        "mvcRenderCommandName", MVCCCommandNames.EDIT_SUBM
SSION);
}
catch (PortalException e) {

    // Set error message.

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

    actionResponse.setRenderParameter(
        "mvcRenderCommandName", MVCCCommandNames.EDIT_SUBM
SSION);
}
}

/**
 * Returns <code>true</code> if we are editing.
 *
 * @param actionRequest
 * @return <code>true</code> if we are editing; <code>false</
code> otherwise

```

```

    */
private boolean isEditAction(ActionRequest actionRequest) {

    return ParamUtil.getString(
        actionRequest, ActionRequest.ACTION_NAME,
        MVCCCommandNames.ADD_SUBMISSION).indexOf(
            MVCCCommandNames.EDIT_SUBMISSION) >= 0;
}

@Reference
protected SubmissionService _submissionService;

@Reference
protected Portal _portal;
}

```



GradeSubmissionMVCActionCommand

```

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

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.exception.SubmissionValidationException;
import com.liferay.training.gradebook.service.SubmissionService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCCommandNames;

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

```

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

/**
 * MVC Command for grading submissions.
 *
 * @author liferay
 *
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
        "mvc.command.name=" + MVCCCommandNames.GRADE_SUBMISSION
    },
    service = MVCActionCommand.class
)
public class GradeSubmissionMVCActionCommand extends BaseMVCActionCommand {

    @Override
    protected void doProcessAction(
        ActionRequest actionRequest, ActionResponse actionResponse)
        throws Exception {
        // Get parameters from request.

        long submissionId = ParamUtil.getLong(actionRequest, "submissionId");
        int grade = ParamUtil.getInteger(actionRequest, "grade");
        String comment = ParamUtil.getString(actionRequest, "comment");

        try {
            // Call the service to grade and comment.

            _submissionService.gradeAndCommentSubmission(
                submissionId, grade, comment);
        }
    }
}
```

```

        // Set success message.

        SessionMessages.add(actionRequest, "submissionGraded"
);

        sendRedirect(actionRequest, actionResponse);
    }
    catch (SubmissionValidationException e) {

        // Set errors.

        e.getErrors().forEach(
            key -> SessionErrors.add(actionRequest, key));
    }

    actionResponse.setRenderParameter(
        "mvcRenderCommandName", MVCCCommandNames.GRADE_SUB
MISSION);
}
}

@Reference
protected SubmissionService _submissionService;
}

```

DeleteSubmissionMVCActionCommand

```

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.SubmissionService;

```

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

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 submissions.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
        "mvc.command.name=" + MVCCommandNames.DELETE_SUBMISSION
    },
    service = MVCActionCommand.class
)
public class DeleteSubmissionMVCActionCommand extends BaseMVCActionCommand {

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

        // Get submission id for the request.

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

        try {

```

```

        // Call the service to delete the submission.

        _submissionService.deleteSubmission(submissionId);

        // Set success message.

        SessionMessages.add(actionRequest, "submissionDeleted"
);

    } catch (PortalException e) {

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

    sendRedirect(actionRequest, actionResponse);
}

@Reference
protected SubmissionService _submissionService;
}

```

Implement the MVC Render Commands

1. **Add** and implement the following three MVC Render Commands in the *gradebook-web* module as follows:

- `com.liferay.training.gradebook.web.portlet.action.EditSubmissionMVCRenderCommand`
- `com.liferay.training.gradebook.web.portlet.action.GradeSubmissionMVCRenderCommand`
- `com.liferay.training.gradebook.web.portlet.action.ViewSubmissionMVCRenderCommand`

EditSubmissionMVCRenderCommand

```
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.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.model.Submission;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.service.SubmissionLocalService;
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.AssignmentPermission;

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 submission edit view.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
```

```
        "mvc.command.name=" + MVCCCommandNames.EDIT_SUBMISSION
    },
    service = MVCRenderCommand.class
)
public class EditSubmissionMVCRenderCommand implements MVCRenderC
ommand {

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

        ThemeDisplay themeDisplay =
            (ThemeDisplay) renderRequest.getAttribute(Web
Keys.THEME_DISPLAY);

        long submissionId =
            ParamUtil.getLong(renderRequest, "submiss
ionId", 0);

        try {

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

            // Notice that "fetch" returns a null instead of exce
ption,
            // when not found.

            Submission submission =
                _submissionLocalService.fetchSubmission(submissio
nId);

            long assignmentId;

            if (submission == null) {
                assignmentId = ParamUtil.getLong(renderRequest, "a
ssignmentId");
            }
            else {
```

```
        assignmentId = submission.getAssignmentId();
    }

    // Call the service to get the assignment.
    // Set the attributes to the request.

    Assignment assignment =
        _assignmentService.getAssignment(assignmentId);

    DateFormat dateFormat = DateFormatFactoryUtil.getSimpleDateFormat(
        "EEEEEE, MMMMM dd, yyyy", renderRequest.getLocale());
    renderRequest.setAttribute("assignment", assignment);
    renderRequest.setAttribute("submission", submission);
    renderRequest.setAttribute("submissionClass", Submission.class);
    renderRequest.setAttribute(
        "assignmentPermission", _assignmentPermission);
    renderRequest.setAttribute(
        "dueDate", dateFormat.format(assignment.getDueDate()));
    renderRequest.setAttribute("backIconVisible", true);

    PortletDisplay portletDisplay = themeDisplay.getPortletDisplay();
    String redirect = renderRequest.getParameter("redirect");

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

    return "/submission/edit_submission.jsp";
}
catch (PortalException e) {
    throw new PortletException(e);
}
```

```
}

@Reference
protected AssignmentPermission _assignmentPermission;

@Reference
private AssignmentService _assignmentService;

@Reference
private SubmissionLocalService _submissionLocalService;
}
```

GradeSubmissionMVCRenderCommand

```
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.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.model.Submission;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.service.SubmissionLocalService;
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.AssignmentPermission;

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 submission grading view.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
        "mvc.command.name=" + MVCCmdNames.GRADE_SUBMISSION
    },
    service = MVCRenderCommand.class
)
public class GradeSubmissionMVCRenderCommand implements MVCRender
Command {

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

        ThemeDisplay themeDisplay =
            (ThemeDisplay) renderRequest.getAttribute(WebKeys.THE
ME_DISPLAY);

        long submissionId = ParamUtil.getLong(renderRequest, "sub
missionId", 0);

        try {

            // Call the service to get the submission for grading.
```

```
// Notice that "fetch" returns a null instead of exception,  
// when not found.  
  
Submission submission =  
    _submissionLocalService.fetchSubmission(submissionId);  
  
long assignmentId;  
  
if (submission == null) {  
    assignmentId = ParamUtil.getLong(renderRequest, "assignmentId");  
}  
else {  
    assignmentId = submission.getAssignmentId();  
}  
  
Assignment assignment =  
    _assignmentService.getAssignment(assignmentId);  
  
DateFormat dateFormat = DateFormatFactoryUtil.getSimpleDateFormat(  
    "EEEEEE, MMMMM dd, yyyy", renderRequest.getLocale());  
  
renderRequest.setAttribute("assignment", assignment);  
renderRequest.setAttribute("submission", submission);  
renderRequest.setAttribute("submissionClass", Submission.class);  
renderRequest.setAttribute(  
    "assignmentPermission", _assignmentPermission);  
  
renderRequest.setAttribute(  
    "createDate", dateFormat.format(submission.getCreateDate()));  
renderRequest.setAttribute(  
    "student", _userLocalService.getUser(  
        submission.getStudentId()).getFullName());
```

```
        renderRequest.setAttribute(
            "dueDate", dateFormat.format(assignment.getDueDat
e()));

        // Set back icon visible.

        PortletDisplay portletDisplay = themeDisplay.getPortl
etDisplay();

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

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

        return "/submission/grade_submission.jsp";
    }
    catch (PortalException e) {
        throw new PortletException(e);
    }
}

@Reference
protected AssignmentPermission _assignmentPermission;

@Reference
private AssignmentService _assignmentService;

@Reference
private SubmissionLocalService _submissionLocalService;

@Reference
private UserLocalService _userLocalService;
}
```

ViewSubmissionMVCRenderCommand

```
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.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.model.Submission;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.service.SubmissionLocalService;
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.AssignmentPermission;

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 single submission view.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
```

```
"javax.portlet.name=" + GradebookPortletKeys.Gradebook,
"mvc.command.name=" + MVCCCommandNames.VIEW_SUBMISSION
},
service = MVCRenderCommand.class
)
public class ViewSubmissionMVCRenderCommand implements MVCRenderC
ommand {

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

    long submissionId = ParamUtil.getLong(renderRequest, "sub
missionId", 0);

    try {

        // Call the service to get the submission for grading.

        // Notice that "fetch" returns a null instead of exce
ption,
        // when not found.

        Submission submission =
            _submissionLocalService.fetchSubmission(submissio
nId);

        long assignmentId = submission.getAssignmentId();

        Assignment assignment =
            _assignmentService.getAssignment(assignmentId);

        DateFormat dateFormat = DateFormatFactoryUtil.getSimpleDateFormat(
            "EEEEEE, MMMMM dd, yyyy", renderRequest.getLocale(
        ));

        // Set attributes to the request.
```

```

        renderRequest.setAttribute("assignment", assignment);
        renderRequest.setAttribute("submission", submission);
        renderRequest.setAttribute("submissionClass", Submission.class);
        renderRequest.setAttribute(
            "assignmentPermission", _assignmentPermission);
        renderRequest.setAttribute(
            "createDate", dateFormat.format(submission.getCreateDate()));
        renderRequest.setAttribute(
            "student", _userLocalService.getUser(
                submission.getStudentId()).getFullName());
        renderRequest.setAttribute(
            "dueDate", dateFormat.format(assignment.getDueDate()));

        // Set back icon visible.

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

        PortletDisplay portletDisplay = themeDisplay.getPortletDisplay();

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

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

        return "/submission/view_submission.jsp";
    }
    catch (PortalException e) {
        throw new PortletException(e);
    }
}

```

@Reference

```

protected AssignmentPermission _assignmentPermission;

@Reference
private AssignmentService _assignmentService;

@Reference
private SubmissionLocalService _submissionLocalService;

@Reference
private UserLocalService _userLocalService;

}

```

Modify the MVC Render Command for Showing a Single Assignment

Modify the MVC Render Command for showing a single assignment so that it includes the list of related submissions in the request attributes:

- Open** the class

```
com.liferay.training.gradebook.web.portlet.action.ViewSingleAssignmentMVCRenderCommand
```

in the *gradebook-web* module.

- Replace** the contents of the class with the following:

```

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

import com.liferay.portal.kernel.dao.search.SearchContainer;
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.model.Submission;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.service.SubmissionService;
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.AssignmentPermission;

import java.text.DateFormat;
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 assignment submissions list view.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + GradebookPortletKeys.Gradebook,
        "mvc.command.name=" + MVCCCommandNames.VIEW_ASSIGNMENT
    },
    service = MVCRenderCommand.class
)
public class ViewSingleAssignmentMVCRenderCommand implements MVCRenderCommand {

    @Override
    public String render(
        RenderRequest renderRequest, RenderResponse renderRespons
```

e)

```
throws PortletException {  
  
    ThemeDisplay themeDisplay =  
        (ThemeDisplay) renderRequest.getAttribute(WebKeys.THE  
ME_DISPLAY);  
  
    long assignmentId = ParamUtil.getLong(renderRequest, "ass  
ignmentId", 0);  
  
    try {  
  
        // Call the service to get the assignment.  
  
        Assignment assignment =  
            _assignmentService.getAssignment(assignmentId);  
  
        DateFormat dateFormat = DateFormatFactoryUtil.getSimpleDateFormat(  
            "EEEEEE, MMMMM dd, yyyy", renderRequest.getLocale());  
  
        // Set attributes to the request.  
  
        renderRequest.setAttribute("assignment", assignment);  
        renderRequest.setAttribute(  
            "dueDate", dateFormat.format(assignment.getDueDat  
e()));  
        renderRequest.setAttribute(  
            "createDate", dateFormat.format(assignment.getCre  
ateDate()));  
  
        // Add submissions list.  
  
        addSubmissionListAttributes(  
            renderRequest, renderResponse, assignmentId);  
  
        // Set permission checker.  
  
        renderRequest.setAttribute(
```

```
        "assignmentPermission", _assignmentPermission);

    // 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 e) {
    throw new PortletException(e);
}
}

/**
 * Adds submission list related attributes to the request.
 *
 * @param renderRequest
 * @throws PortalException
 */
private void addSubmissionListAttributes(
    RenderRequest renderRequest, RenderResponse renderResponse,
    long assignmentId) throws PortalException {

    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;

// Call the service to get the list of submissions.
// Notice that the search only targets to submissionText
field.

List<Submission> submissions =
    _submissionService.getSubmissionsByAssignment(
        themeDisplay.getScopeGroupId(), assignmentId, sta
rt,
        end);

long submissionsCount =
    _submissionService.getSubmissionsCountByAssignment(
        themeDisplay.getScopeGroupId(), assignmentId);

// Set request attributes.

renderRequest.setAttribute(
    "canAddSubmission", canAddSubmission(submissions, ren
derRequest));
    renderRequest.setAttribute("submissions", submissions);
    renderRequest.setAttribute("submissionsCount", submission
sCount);
}

/**
 * Returns <code>true</code> if user is allowed to do subm
it.
*
* @param submissions
```

```
*           list of submissions
* @param renderRequest
* @return
* @throws PortalException
*/
private boolean canAddSubmission(
    List<Submission> submissions, RenderRequest renderRequest)

throws PortalException {

    ThemeDisplay themeDisplay =
        (ThemeDisplay) renderRequest.getAttribute(WebKeys.THE
ME_DISPLAY);

    long userId = themeDisplay.getUserId();

    boolean hasAlreadySubmitted = false;

    if (submissions != null) {
        for (Submission submission : submissions) {
            if (submission.getStudentId() == userId) {
                hasAlreadySubmitted = true;
                break;
            }
        }
    }

    if (!hasAlreadySubmitted) {
        return true;
    }

    return false;
}

@Reference
protected AssignmentPermission _assignmentPermission;

@Reference
private AssignmentService _assignmentService;

@Reference
```

```

    private Portal _portal;

    @Reference
    private SubmissionService _submissionService;

    @Reference
    private UserLocalService _userLocalService;
}
```

Implement the JSP files

1. **Create** a folder `src/main/resources/META-INF/resources/submission` in the *gradebook-web* module.
2. **Implement** the following JSP files in the folder:

edit_submission.jsp

```

<%-- 
    Submission editing view.
--%>

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

<%-- Error messages. --%>

<liferay-ui:error key="serviceErrorDetails">
    <liferay-ui:message key="error.assignment-service-error" arguments='<%= SessionErrors.get(liferayPortletRequest, "serviceError Details") %>' />
</liferay-ui:error>
<liferay-ui:error key="submissionTooLate" message="error.submission-is-too-late" />
<liferay-ui:error key="onlyOneSubmissionAllowed" message="error.only-one-submission-allowed" />
<liferay-ui:error key="submissionTextNull" message="error.submission-text-null" />

<%-- Generate cancel URL. --%>
```

```

<portlet:renderURL portletMode="view" var="assignmentsURL" />

<portlet:renderURL var="cancelURL">
    <portlet:param name="mvcRenderCommandName" value="<%=>MVCCoomma
ndNames.VIEW_ASSIGNMENT %>" />
    <portlet:param name="redirect" value="${assignmentsURL}" />
    <portlet:param name="assignmentId" value="${assignment.getAss
ignmentId()}" />
</portlet:renderURL>

<%-- Generate add / edit action URL. --%>

<c:choose>

    <c:when test="${not empty submission}">

        <portlet:actionURL var="submissionActionURL" name="<%=>MVC
CommandNames.EDIT_SUBMISSION %>">
            <portlet:param name="redirect" value="${param.redirec
t}" />
        </portlet:actionURL>

        <c:set var="editTitle" value="edit-submission-for-x"/>

    </c:when>

    <c:otherwise>

        <portlet:actionURL var="submissionActionURL" name="<%=>MVC
CommandNames.ADD_SUBMISSION %>">
            <portlet:param name="redirect" value="${param.redirec
t}" />
        </portlet:actionURL>

        <c:set var="editTitle" value="add-submission-for-x"/>

    </c:otherwise>
</c:choose>

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

```

```
<h1><liferay-ui:message key="${editTitle}" arguments="${assignment.getTitle(locale)}"/></h1>

<%--Show editing control based on user's permissions. --%>

<c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignmentId, 'EDIT_SUBMISSION')}">

    <aui:model-context bean="${submission}" model="${submissionClass}" />

    <aui:form action="${submissionActionURL}" name="fm"
        onSubmit='<%= "event.preventDefault(); " + renderResponse.getNamespace() + "saveSubmission();" %>'>

        <aui:input
            name="assignmentId"
            type="hidden"
            value="${assignment.assignmentId}">
            />
        <aui:input name="submissionId" field="submissionId" type="hidden" />

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

            <aui:fieldset>

                <aui:field-wrapper required="true">
                    <label for="submissionText">
                        <liferay-ui:message key="your-submission" />
                    </label>
                    <liferay-ui:input-editor
                        contents="${submission.submissionText}"
                        editorName='alloyeditor'
                        name="submissionTextEditor"
                        placeholder="submission-text" />
                </aui:field-wrapper>
            </aui:fieldset>
        </aui:fieldset-group>
    </aui:form>
</c:if>
```

```

                <aui:input name="submissionText" type="hi
dden" />
                </aui:field-wrapper>
            </aui:fieldset>
        </aui:fieldset-group>

<%--Buttons. --%>

<aui:button-row>
    <aui:button cssClass="btn btn-primary" type="subm
it" />
    <aui:button cssClass="btn btn-secondary" onClick=
"<%=cancelURL %>" type="cancel" />
</aui:button-row>
</aui:form>

<aui:script>

    /**
     * Set editor value to the hidden field which transpor
ts the value to the backend.
    */
    function <portlet:namespace />saveSubmission() {

        var editorValue = window['<portlet:namespace />su
bmissionTextEditor'].getHTML();

        window['<portlet:namespace />submissionText'].val
ue = editorValue;

        submitForm(document.<portlet:namespace />fm);
    }

</aui:script>
</c:if>
</div>

```

entry_actions.jsp

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

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

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

    <%-- View action. Show for the owner for those having grading
permissions. --%>

    <c:if test="${submission.studentId eq user.userId or assignmentPermission.contains(permissionChecker, assignment.assignmentId,
'GRADE_SUBMISSION' )}">
        <portlet:renderURL var="viewSubmissionURL">
            <portlet:param name="mvcRenderCommandName" value="<%=
MVCCommandNames.VIEW_SUBMISSION %>" />
            <portlet:param name="redirect" value="${currentURL}" />
            <portlet:param name="submissionId" value="${submission.submissionId}" />
        </portlet:renderURL>

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

    <%-- Grade action. --%>

    <c:if test="${assignmentPermission.contains(permissionChecker,
, assignment.assignmentId, 'GRADE_SUBMISSION' )}">
        <portlet:renderURL var="gradeSubmissionURL">
            <portlet:param name="mvcRenderCommandName" value="<%=
MVCCommandNames.GRADE_SUBMISSION %>" />
            <portlet:param name="redirect" value="${currentURL}" />
            <portlet:param name="submissionId" value="${submission.submissionId}" />
        </portlet:renderURL>
    </c:if>

```

```
        <liferay-ui:icon message="grade" url="\${gradeSubmissionUR
L}" />
    </c:if>

    <%-- Edit action. Don't allow editing graded submission. --%>

    <c:if
        test="\${submission.grade lt 0
            and (assignmentPermission.contains(permissionChecker,
assignment.assignmentId, 'EDIT_SUBMISSION')
            or submission.studentId eq user.userId)}">

        <portlet:renderURL var="editSubmissionURL">
            <portlet:param name="mvcRenderCommandName" value="<%=
MVCCommandNames.EDIT_SUBMISSION %>" />
            <portlet:param name="redirect" value="\${currentURL}" />
            <portlet:param name="submissionId" value="\${submissio
n.submissionId}" />
        </portlet:renderURL>

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

    <%-- Delete action. --%>

    <c:if test="\${assignmentPermission.contains(permissionChecker,
, assignment.assignmentId, 'DELETE_SUBMISSION' ) or submission.s
tudentId eq user.userId}">

        <portlet:actionURL var="deleteSubmissionURL" name="<%=
MVC
CommandNames.DELETE_SUBMISSION %>">
            <portlet:param name="redirect" value="\${currentURL}" />
            <portlet:param name="submissionId" value="\${submissio
n.submissionId}" />
        </portlet:actionURL>

        <liferay-ui:icon-delete message="delete" url="\${deleteSub
```

```
missionURL}" confirmation="are-you-sure-to-delete" />
</c:if>

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

entry_search_columns.jspf

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

<portlet:renderURL var="viewSubmissionURL">
    <portlet:param name="mvcRenderCommandName" value="<%=MVCComma
ndNames.VIEW_SUBMISSION %>" />
    <portlet:param name="redirect" value="${currentURL}" />
    <portlet:param name="submissionId" value="${entry.submissionI
d}" />
</portlet:renderURL>

<%-- Submission ID. --%>

<liferay-ui:search-container-column-text
    href="${viewSubmissionURL}"
    name="id"
    value="${entry.submissionId}"
/>

<%-- Student. --%>

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

<%-- Submit date. --%>

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

```

<%-- Grade --%>

<liferay-ui:search-container-column-text name="grade">
    <c:choose>
        <c:when test="${entry.grade lt 0}">
            <liferay-ui:message key="submission-not-graded" />
        </c:when>
        <c:otherwise>
            <fmt:formatNumber value="${entry.grade}" type="number"
        />
        </c:otherwise>
    </c:choose>
</liferay-ui:search-container-column-text>

<%-- Actions menu. --%>

<liferay-ui:search-container-column-jsp
    name="actions"
    path="/submission/entry_actions.jsp"
/>

```



grade_submission.jsp

```

<%--
    Single submission view.
--%>

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

<%--Grading action url. --%>

<portlet:actionURL name="<%=MVCCCommandNames.GRADE_SUBMISSION %>"
var="submissionGradingURL">
    <portlet:param name="redirect" value="${param.redirect}" />
</portlet:actionURL>

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

```

```
<h1><liferay-ui:message key="submission-information" /></h1>

<div class="submission-metadata">

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

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

        <dt><liferay-ui:message key="grade" /></dt>
        <dd>
            <c:choose>
                <c:when test="${submission.grade lt 0}">
                    <i><liferay-ui:message key="not-graded" />
                </i>
            </c:when>
            </c:otherwise>
                <fmt:formatNumber value="${submission.gra
de}" type="number" />
            </c:otherwise>
        </c:choose>
        </dd>
        <dt><liferay-ui:message key="submission-comment" /></
dt>
        <dd>
            <c:choose>
                <c:when test="${empty submission.comment}">
                    <i><liferay-ui:message key="no-comment" />
                </i>
            </c:when>
            <c:otherwise>
                ${submission.comment}
            </c:otherwise>
        </c:choose>
        </dd>
    </dl>
</div>
```

```
<h2><liferay-ui:message key="submission-text" /></h2>

<div class="submission-text">
    ${submission.submissionText}
</div>

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

    <aui:form action="${submissionGradingURL}" name="gradingForm" onSubmit="event.preventDefault();">

        <aui:input name="assignmentId" value="${submission.assignmentId}" type="hidden" />
        <aui:input name="submissionId" value="${submission.submissionId}" type="hidden" />

        <aui:fieldset-group markupView="lexicon">
            <aui:fieldset>
                <aui:field-wrapper required="true">
                    <label for="grade"><liferay-ui:message key="grade-submission" /></label>
                    <aui:input name="grade" value="${submission.grade}" min="4" max="10" type="number" />
                </aui:field-wrapper>
            </aui:fieldset>
            <aui:fieldset>
                <aui:field-wrapper required="true">
                    <label for="comment"><liferay-ui:message key="submission-comment" /></label>
                    <liferay-ui:input-editor contents="${submission.comment}" editorName="alloyeditor" name="commentEditor" placeholder="comment" />
                    <aui:input name="comment" type="hidden" />
                </aui:field-wrapper>
            </aui:fieldset>
        </aui:fieldset-group>
    </aui:form>
```

```

        </aui:fieldset-group>
            <aui:button cssClass="btn btn-primary" type="submit" />
            <aui:button cssClass="btn btn-secondary" onClick="${param.redirect}" type="cancel" />
        </aui:form>

        <aui:script>

        /**
         * Handle form submit. Set editor value to the hidden field
         * which transports the value to the backend.
         */
        AUI().ready(function() {

            $('#<portlet:namespace />gradingForm').on('submit', function() {

                var editorValue = window['<portlet:namespace />commentEditor'].getHTML();

                window['<portlet:namespace />comment'].value = editorValue;

                submitForm(document.<portlet:namespace />gradingForm);
            });
        });

        </aui:script>
    </div>

```



view_submission.jsp

```

<%-- 
    Single submission view.
--%>

```

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

<%-- Success messages. --%>

<liferay-ui:success key="submissionGraded" message="submission-graded-successfully" />

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

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

    <div class="submission-metadata">

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

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

            <dt><liferay-ui:message key="grade" /></dt>
            <dd>
                <c:choose>
                    <c:when test="${submission.grade lt 0}">
                        <i><liferay-ui:message key="not-graded" />
                    </i>
                </c:when>
                </c:otherwise>
                    <fmt:formatNumber value="${submission.grade}" type="number" />
                </c:otherwise>
            </c:choose>
        </dd>
        <dt><liferay-ui:message key="submission-comment" /></dt>
        <dd>
            <c:choose>
                <c:when test="${empty submission.comment}">
                    <i><liferay-ui:message key="no-comment" />
                </c:when>
```

```

</i>
    </c:when>
    <c:otherwise>
        ${submission.comment}
    </c:otherwise>
</c:choose>
</dd>
</dl>
</div>

<h2><liferay-ui:message key="submission-text" /></h2>

<div class="submission-text">
    ${submission.submissionText}
</div>

<aui:button-row>

    <%-- Show edit button if permissions allow and submissions
is not graded. --%>

    <c:if
        test="${submission.grade lt 0
            and (assignmentPermission.contains(permissionChec
ker, assignment.assignmentId, 'DELETE_SUBMISSION')
            or submission.studentId eq user.userId)}">

        <portlet:renderURL var="editSubmissionURL">
            <portlet:param name="mvcRenderCommandName" value=
"${MVCCommandNames.EDIT_SUBMISSION}" />
            <portlet:param name="redirect" value="${currentUR
L}" />
            <portlet:param name="submissionId" value="${submi
ssion.submissionId}" />
        </portlet:renderURL>

        <aui:button cssClass="btn btn-primary" onClick="${edi
tSubmissionURL}" value="edit" />
    </c:if>

```

```

<%--Show grading button. --%>

<c:if
    test="${assignmentPermission.contains(
        permissionChecker, assignment.assignmentId, 'GRADE_SUBMISSION' )}">

    <portlet:renderURL var="gradeSubmissionURL">
        <portlet:param name="mvcRenderCommandName" value=
"\<%=\!MVCCommandNames.GRADE_SUBMISSION %>" />
        <portlet:param name="redirect" value="${currentURL}" />
        <portlet:param name="submissionId" value="${submission.submissionId}" />
    </portlet:renderURL>

    <aui:button cssClass="btn btn-primary" onClick="${gradeSubmissionURL}" value="grade" />
</c:if>

    <aui:button cssClass="btn btn-secondary" onClick="${param.redirect}" type="cancel" />
</aui:button-row>
</div>

```

Customize the JSP for Showing a Single Assignment

Implement showing the list of submissions on single Assignment view:

- Open** the JSP file `src/main/resources/META-INF/resources/assignment/view_assignment.jsp`
- Replace** the contents with the following. See the highlighted lines for changes:

```

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

<liferay-ui:success key="assignmentUpdated" message="assignment-updated-successfully" />
<liferay-ui:success key="submissionAdded" message="submission-added" />

```

```
ed-successfully" />
<liferay-ui:success key="submissionDeleted" message="submission-d
eleted-successfully" />
<liferay-ui:success key="submissionGraded" message="submission-gr
aded-successfully" />

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

    <h1>${assignment.getTitle(locale)}</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.getDescription(locale)}</dd>
        </dl>
    </div>

    <!-- Submissions -->

    <c:choose>
        <c:when test="${assignmentPermission.contains(permissionC
hecker, assignment.assignmentId, 'VIEW_SUBMISSIONS')}">

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

            <%-- The search container. --%>

```

```
<liferay-ui:search-container
    emptyResultsMessage="submissions"
    id="submissionEntries"
    iteratorURL="${portletURL}"
    total ="${submissionCount}">

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

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

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

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

    <liferay-ui:search-iterator markupView="lexicon"
/>

</liferay-ui:search-container>

<c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignmentId, 'ADD_SUBMISSION')}">
    <portlet:renderURL var="addSubmissionURL">
        <portlet:param name="mvcRenderCommandName" value="<%=MVCCCommandNames.EDIT_SUBMISSION %>" />
        <portlet:param name="redirect" value ="${currentURL}" />
        <portlet:param name="assignmentId" value ="${assignment.assignmentId}" />
    </portlet:renderURL>
    <br />
    <p><a href ="${addSubmissionURL}"><liferay-ui:message key="add-submission" /></a></p>
</c:if>
```

```

</c:when>
<c:otherwise>

    <clay:alert
        message='<%=LanguageUtil.get(request, "viewing-submissions-not-allowed") %>'
        title="Info" />
    </c:otherwise>
</c:choose>
</div>

```

Test the Application

Be sure that you have rebuilt the services. If you encounter `ClassNotFoundException` while testing, remove all the Gradebook modules from the server and redeploy to refresh the resources.

1. **Sign in** to the portal with your web browser.
2. **Open** the Gradebook Application.
3. **Create** or open an Assignment.
4. **Try** creating a submission from the link on the bottom of the page.
5. **Try** editing, grading and deleting submissions.

<
GRADEBOOK

My First Assignment

Assignment Information

Created
Tuesday, April 09, 2019

Created By
test

Assignment Due Date
Tuesday, April 30, 2019

Description
This is it finally.

Submissions

ID	Student	Submitted	Grade	Actions
201	 Liferay Demo	20 Seconds Ago	9	⋮

[Add New Submission](#)

Optional Exercise

Add JavaScript Resources

Exercise Goals

- Initialize the `package.json` file for the *gradebook-web* module
- Install the Liferay NPM Bundler
- Install the Tooltip library
- Install the Event Emitter library
- Add the NPM Bundler build script
- Test the User Interface

The exercise requires the [Node.js](#) version 6.11.0 or greater to be installed in your development environment.

Initialize the `package.json` file

1. **Open** the command prompt and navigate to the root folder of *gradebook-web* project.
2. **Run** the following command to initialize the `package.json` :

```
npm init -y
```

The generated file will look like this:

```
{
  "name": "gradebook-web",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  }
}
```

```
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Install the Liferay NPM Bundler

Liferay NPM Bundler takes care of formatting the NPM bundles automatically for Liferay AMD loader. We need to include the NPM package for the tool:

1. **Open** the command prompt and navigate to the root folder of *gradebook-web* project.
2. **Run** the following command:

```
npm install --save-dev liferay-npm-bundler
```

Install the Tooltip Library

Install the Tooltip library (<https://www.npmjs.com/package/tooltip>):

1. **Run** the following command:

```
npm install tooltip --save
```

Install the Event Emitter Library

We also need to install Node's event emitter:

1. **Run** the following command:

```
npm install events --save
```

Modify the `package.json` Build Scripts

Add the liferay-npm-bundler to the `package.json` build script to pack the needed NPM packages and transform them to AMD. Remove the test script:

1. **Open** the `package.json` in the root module of *gradebook-web*.
2. **Replace** the contents with the following (see the highlighted lines for changes):

```
{
  "name": "gradebook-web",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "liferay-npm-bundler"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "liferay-npm-bundler": "^2.8.0"
  },
  "dependencies": {
    "events": "^3.0.0",
    "tooltip": "^1.6.1"
  }
}
```

Add the Tooltip to `view.jsp`

Add the placeholder for the tooltip in the JSP file:

1. Open the `src/main/resources/META-INF/resources/view.jsp` file.
2. Add just before the assignment heading:

```
<p>
  <a class="gradebook-tip" href="javascript:void(0);"
      data-tooltip=<liferay-ui:message key="assignments-help-text" />">
    <liferay-ui:message key="help" />
    <clay:icon symbol="question-circle" />
  </a>
</p>
```

3. Add the Javascript to the end of the file:

```
<aui:script>
  Liferay.Loader.require('gradebook-web$tooltip@1.6.1/dist/Tool
```

```

tip', function(tooltip) {
    tooltip();
});
</aui:script>

```

The complete file will now look like:

```

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

<liferay-ui:error key="serviceErrorDetails">
    <liferay-ui:message arguments='<%= SessionErrors.get(liferayPortletRequest, "serviceErrorDetails") %>' key="error.assignment-service-error" />
</liferay-ui:error>
<liferay-ui:success key="assignmentAdded" message="assignment-added-successfully" />
<liferay-ui:success key="assignmentUpdated" message="assignment-updated-successfully" />
<liferay-ui:success key="assignmentDeleted" message="assignment-deleted-successfully" />

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

    <p>
        <a class="gradebook-tip" href="javascript:void(0);"
            data-tooltip="<liferay-ui:message key="assignments-help-text" />">
            <liferay-ui:message key="help" />
            <clay:icon symbol="question-circle" />
        </a>
    </p>

    <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="${assignmentsManagementToolbarDisplayStyle()}"
            markupView="lexicon"
        />
    </liferay-ui:search-container>
</div>

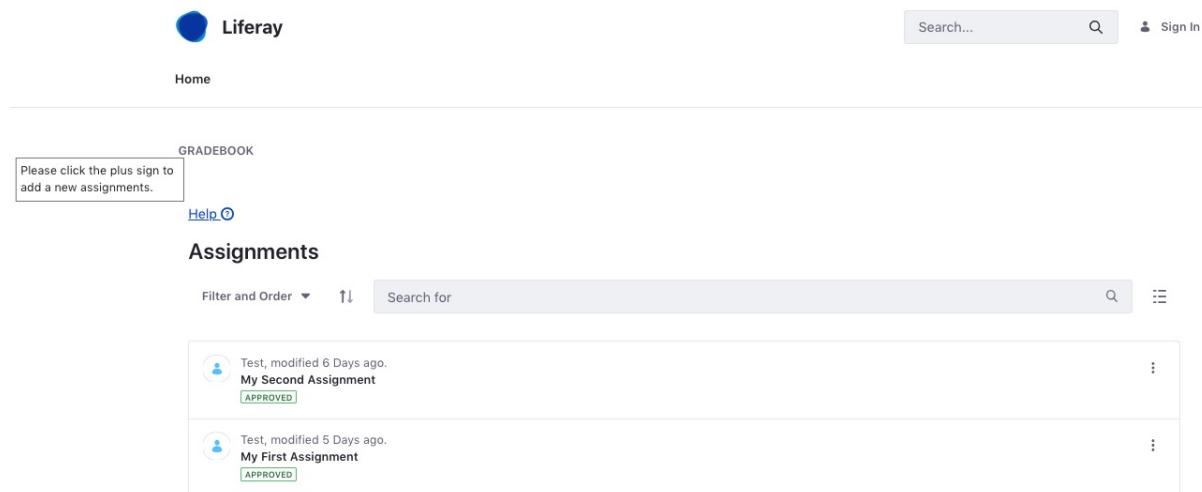
<aui:script>
    Liferay.Loader.require('gradebook-web$tooltip@1.6.1/dist/Tooltip',
        function(tooltip) {
            tooltip();
    }
</aui:script>
```

```
});  
</aui:script>
```

Note that the Tooltip library might be different from 1.6.1, used in this exercise. Check the library version in `package.json` file in the `gradebook-web` root folder and update in the code, if necessary

Test the User Interface

1. **Open** the Gradebook application in your browser
2. **Hover** your cursor over the help / question mark icon.



The screenshot shows the Liferay Gradebook application interface. At the top, there is a navigation bar with the Liferay logo, a search bar, and a sign-in link. Below the navigation bar, the page title is "Home". A "GRADEBOOK" portlet is visible, containing a message: "Please click the plus sign to add a new assignments." Below this message is a "Help" link with a question mark icon. The main content area is titled "Assignments". It features a table with two rows of assignment data. Each row includes a user icon, the assignment name, a timestamp, and a status indicator labeled "APPROVED". There are also three-dot and search/filter icons at the top of the table.

Test, modified 6 Days ago.	My Second Assignment	APPROVED	⋮
Test, modified 5 Days ago.	My First Assignment	APPROVED	⋮

See the Developer Network (https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/using-npm-in-your-portlets) for more information about using the NPM in your portlet modules.

Optional Exercise

Make the Gradebook Configurable

Exercise Goals

- Declare dependencies in the `build.gradle`
- Create a configuration interface in the API module
- Add the configuration interface package to the exported packages in the `bnd.bnd`
- Add localization resources for the configuration
- Test the configuration user interface
- Implement configuration support to the Assignment validator service
- Show messages in the user interface
- Add new error message localization resources
- Test the Application

Declare Dependencies

We need to add dependencies for the BND annotations and Metatype API:

1. Open the `build.gradle` in the *gradebook-api* module.
2. Add the following dependencies:

```
compileOnly group: "biz.aQute.bnd", name: "biz.aQute.bnd.annotation", version: "3.1.0"
compileOnly group: "com.liferay", name: "com.liferay.portal.configuration.metatype.api"
```

Create the Configuration Interface in the API Module

1. Create a new interface

`com.liferay.training.gradebook.configuration.GradebookSystemServiceConfiguration` in the *gradebook-api* module.

2. **Implement** the interface as follows:

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

import com.liferay.portal.configuration.metatype.annotations.ExtendedObjectClassDefinition;

import aQute.bnd.annotation.metatype.Meta;

/**
 * Configuration interface for Gradebook service.
 *
 * An user interface for this interface is automatically created
 * in Control Panel -> System settings.
 *
 * @see <a href="https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-2/making-applications-configurable">Tutorial
 *      on making configurable applications at Liferay Developer Network</a>
 * @author liferay
 */

@ExtendedObjectClassDefinition(
    category = "Gradebook",
    scope = ExtendedObjectClassDefinition.Scope.SYSTEM
)
@Meta.OCD(
    id = "com.liferay.training.gradebook.configuration.GradebookSystemServiceConfiguration",
    localization = "content/Language",
    name = "gradebook-service-configuration-name"
)
public interface GradebookSystemServiceConfiguration {

    @Meta.AD(
        deflt = "10",
        ...
    )
}
```

```

        description = "description-min-length-description",
        name = "description-min-length-name",
        required = false
    )
    public int descriptionMinLength();

    @Meta.AD(
        deflt = "200",
        description = "description-max-length-description",
        name = "description-max-length-name",
        required = false
    )
    public int descriptionMaxLength();
}

```

The naming syntax of the interface is by convention [Application][Scope] [Layer]Configuration.

Note that the configuration ID **has to be** the fully qualified name of the interface.

Add Exported Package

To be able to consume the configuration from other bundles, we have to expose it by exporting its package:

1. Open the `bnd.bnd` file of the *gradebook-api* module.
2. Implement the export as follows (highlighted code):

```

Bundle-Name: gradebook-api
Bundle-SymbolicName: com.liferay.training.gradebook.api
Bundle-Version: 1.0.0
Export-Package: \
    com.liferay.training.gradebook.configuration,\
    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
-Check: EXPORTS

```

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

Add the Localization Resources

Notice the resource bundle property in our configuration interface:

```
@Meta.OCD(  
    id = "com.liferay.training.gradebook.configuration.Gradebooks  
systemServiceConfiguration",  
    localization = "content/Language",  
    name = "gradebook-service-configuration-name"  
)
```

Let's add the referenced resources to localize the user interface:

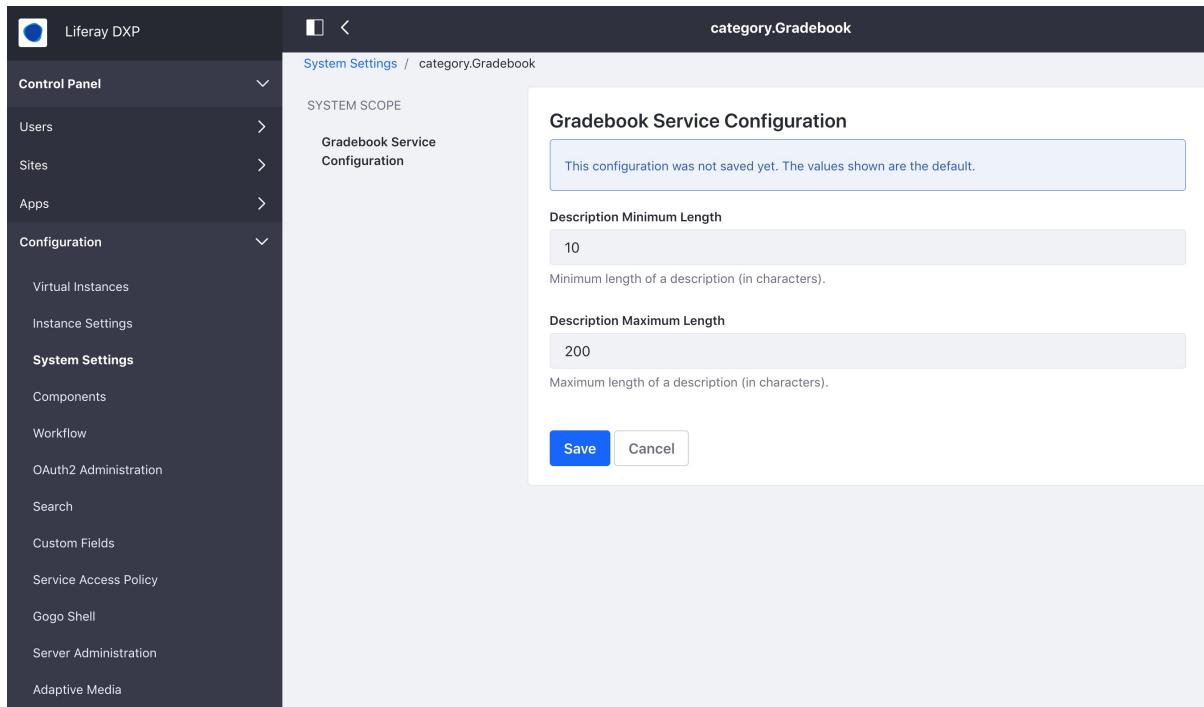
1. **Create** a file `src/main/resources/content/Language.properties` in the *gradebook-api* module
2. **Implement** the file as follows:

```
gradebook-service-configuration-name=Gradebook Service Configu  
ration  
description-max-length-description=Maximum length of a descrip  
tion (in characters).  
description-max-length-name=Description Maximum Length  
description-min-length-description=Minimum length of a descrip  
tion (in characters).  
description-min-length-name=Description Minimum Length
```

Test the Configuration Interface

Creating a configuration interface automatically generates an user interface.

1. **Open** the browser and go to the *Control Panel* → *Configuration* → *System Settings*.
You should see the Gradebook configuration in the *Other* category. You can also search for *Gradebook*.
2. **Click** the configuration icon.



Declare Dependencies

We need to add a dependency to the Metatype API:

1. Open the `build.gradle` in the `gradebook-service` module.
2. Add the following dependency:

```
compileOnly group: "com.liferay", name: "com.liferay.portal.co
nfiguration.metatype.api"
```

Implement Configuration Support to the Assignment Validator Service

1. Open the validator service component

```
com.liferay.training.gradebook.util.validator.AssignmentValidato
rImpl
```

in the `gradebook-service` module.

2. Add the `configurationPid` component property as follows:

```
@Component(
    configurationPid = "com.liferay.training.gradebook.config
uration.GradebookSystemServiceConfiguration",
    immediate = true,
    service = AssignmentValidator.class
)
```

3. **Add** a volatile variable to the end of the class for holding the configuration:

```
private volatile GradebookSystemServiceConfiguration _moduleConfiguration;
```

4. **Add** a component activation method to instantiate the configuration:

```
@Activate  
@Modified  
private void activate(Map<String, Object> properties) {  
  
    _moduleConfiguration = ConfigurableUtil.createConfigurable(  
        GradebookSystemServiceConfiguration.class, properties  
    );  
}
```

5. **Replace** the contents of `isDescriptionValid()` method with the following (notice the highlighted code):

```
private boolean isDescriptionValid(  
    final Map<Locale, String> descriptionMap, final List<String> errors) {  
  
    boolean result = true;  
  
    // Verify the map has something  
  
    if (MapUtil.isEmpty(descriptionMap)) {  
        errors.add("assignmentDescriptionEmpty");  
        result = false;  
    }  
    else {  
  
        // Get the default locale  
  
        Locale defaultLocale = LocaleUtil.getSiteDefault();
```

```
        String descriptionHTML = descriptionMap.get(defaultLocale);

        if ((Validator.isBlank(descriptionHTML))) {
            errors.add("assignmentDescriptionEmpty");
            result = false;
        }

        // Strip HTML tags from text.

        String descriptionText = HtmlUtil.stripHtml(descriptionHTML);

        if (Validator.isBlank(descriptionText)) {
            errors.add("assignmentDescriptionEmpty");
            result = false;
        }

        if (descriptionText.length() < _moduleConfiguration.descriptionMinLength()) {
            errors.add("assignmentDescriptionTooShort");
            result = false;
        }

        else if (descriptionText.length() > _moduleConfiguration.descriptionMaxLength()) {
            errors.add("assignmentDescriptionTooLong");
            result = false;
        }

    }

    return result;
}
```

6. **Organize** missing imports.

Show Messages in the User Interface

1. **Open** the file `src/main/resources/META-INF/resources/assignment/edit_assignment.jsp` in *gradebook-web* module.
2. **Add** the following messages after you find the error message tags in the beginning of the file:

```
<liferay-ui:error key="assignmentDescriptionTooShort" message="error.description-too-short" />
<liferay-ui:error key="assignmentDescriptionTooLong" message="error.description-too-long" />
```

Add Error Message Localizations

1. **Open** the localization file `src/main/resources/content/Language.properties` in the *gradebook-web* module
2. **Add** localizations for the new error messages:

```
error.description-too-short=Description text too short.
error.description-too-long=Description text too long.
```

Test the Application

1. **Test** creating new Assignments with either too short or too long descriptions, after refreshing.

GRADEBOOK

◀

ⓘ Error: Description text too short.

Add New Assignment

Title *

 en-US

Description *

Due Date *

04/09/201905:55 PM

Save Cancel

Optional Exercise

Enable Workflows for Assignments

Exercise Goals

- Add WorkflowInstance reference to `service.xml`
- Manage WorkflowInstance resources in the Assignment local service
- Create an Assignment workflow handler
- Implement support for status in the getter methods
- Implement support for workflow status in the `ViewAssignmentsMVCRenderCommand`
- Implement support for workflow status in the JSP files
- Test the application

This exercise requires that you have completed the "Integrate with the Asset Framework" exercise.

Add WorkflowInstance Reference to `service.xml`

Remember how in the *Integrate with the Asset Framework* exercise we added the status fields to the Assignment entity. These fields are also required in integrating with the Workflow framework.

We'll make the `WorkflowInstanceLink` service , which is responsible for creating workflow resources for model entities, available in the Assignment local service:

1. Open the `service.xml` in the *gradebook-service* module.
2. Add `WorkflowInstanceLink` to the Assignment entity's references as follows:

```
<reference entity="WorkflowInstanceLink" package-path="com.liferay.portal" />
```

3. **Rebuild** the service.

Manage WorkflowInstanceLink resources in the Assignment Local Service

Workflows are bound to model entities with WorkflowInstanceLink resources. Like with permission and Asset resources, we have to take care of managing these resources in the Assignment local service:

1. **Open** the class

```
com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl
```

in the *gradebook-service* module.

2. **Implement** a new method for creating a WorkflowInstanceLink as follows:

```
protected Assignment startWorkflowInstance(
    long userId, Assignment assignment, ServiceContext serviceContext)
    throws PortalException {

    Map<String, Serializable> workflowContext = new HashMap();
;

    String userPortraitURL = StringPool.BLANK;
    String userURL = StringPool.BLANK;

    if (serviceContext.getThemeDisplay() != null) {
        User user = userLocalService.getUser(userId);

        userPortraitURL =
            user.getPortraitURL(serviceContext.getThemeDisplay());
        userURL = user.getDisplayURL(serviceContext.getThemeDisplay());
    }

    workflowContext.put(
        WorkflowConstants.CONTEXT_USER_PORTrait_URL, userPortraitURL);
    workflowContext.put(WorkflowConstants.CONTEXT_USER_URL, u
```

```

    serURL);

        return WorkflowHandlerRegistryUtil.startWorkflowInstance(
            assignment.getCompanyId(), assignment.getGroupId(), u
serId,
            Assignment.class.getName(), assignment.getAssignmentI
d(),
            assignment, serviceContext, workflowContext);
    }
}

```

3. Organize missing imports.

Next, implement updating the status fields and managing WorkFlowInstances on creating and deleting Assignments:

1. **Implement** setting status fields and creating a WorkFlowInstanceLink in the `addAssignment()` method. Replace the method's code with the following. See the highlighted lines for changes:

```

public Assignment addAssignment(long groupId, Map<Locale, Str
ing> titleMap, Map<Locale, String> descriptionMap,
        Date dueDate, ServiceContext serviceContext) throws P
ortalException {

    // Validate assignment parameters.

    _assignmentValidator.validate(titleMap, descriptionMap, d
ueDate);

    // 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 assignment. 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.setDescriptionMap(descriptionMap);
assignment.setGroupId(groupId);
assignment.setModifiedDate(serviceContext.getModifiedDate(new Date()));
assignment.setTitleMap(titleMap);
assignment.setUserId(userId);
assignment.setUserName(user.getScreenName());

// Set Status fields.

assignment.setStatus(WorkflowConstants.STATUS_DRAFT);
assignment.setStatusByUserId(userId);
assignment.setStatusByUserName(user.getFullName());
assignment.setStatusDate(serviceContext.getModifiedDate(null));

// 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.getName(),
```

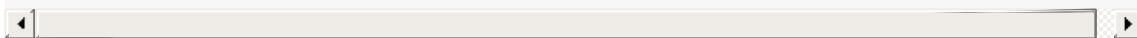
```
        assignment.getAssignmentId(), portletActions, add
GroupPermissions, addGuestPermissions);

        // Update asset.

        updateAsset(assignment, serviceContext);

        // Start workflow instance and return the assignment.

        return startWorkflowInstance(userId, assignment, serviceC
ontext);
    }
```



2. **Implement** deleting the WorkflowInstanceLink in `deleteAssignment()`. Replace the code with following. See highlighted rows for changes:

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

    // Delete permission resources.

    resourceLocalService.deleteResource(
        assignment, ResourceConstants.SCOPe_INDIVIDUAL);

    // Delete the Asset resource.

    assetEntryLocalService.deleteEntry(
        Assignment.class.getName(), assignment.getAssignmentI
d());

    // Delete the workflow resource.

    workflowInstanceLinkLocalService.deleteWorkflowInstanceLi
nks(
        assignment.getCompanyId(), assignment.getGroupId(),
        Assignment.class.getName(), assignment.getAssignmentI
d());
```

```
// Delete the Assignment

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

3. **Organize** imports.

Create an Assignment Workflow Handler

A workflow handler is an OSGi component that registers itself to the OSGi service registry as responsible for handling workflow status changes on defined model entities.

For updating the workflow status, create first the actual worker method in the local service. This method should also sync the `visible` field of entity's Asset resource.

1. **Open** the class

```
com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl
```

`in the gradebook-service module.`

2. **Implement** a new `updateStatus()` method as follows:

```
public Assignment updateStatus(
    long userId, long assignmentId, int status,
    ServiceContext serviceContext)
    throws PortalException, SystemException {

    User user = userLocalService.getUser(userId);
    Assignment assignment = getAssignment(assignmentId);

    assignment.setStatus(status);
    assignment.setStatusByUserId(userId);
    assignment.setStatusByUserName(user.getFullName());
    assignment.setStatusDate(new Date());

    assignmentPersistence.update(assignment);

    if (status == WorkflowConstants.STATUS_APPROVED) {

        assetEntryLocalService.updateVisible(
            Assignment.class.getName(), assignmentId, true);
    }
}
```

```
    }

    else {

        assetEntryLocalService.updateVisible(
            Assignment.class.getName(), assignmentId, false);
    }

    return assignment;
}
```

3. **Organize** missing imports.

4. **Rebuild** the service.

Now implement the WorkflowHandler component:

1. **Create** a class

com.liferay.training.gradebook.service.workflow.AssignmentWorkflowHandler in the *gradebook-service* module.

2. **Implement** the class as follows:

```
package com.liferay.training.gradebook.service.workflow;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.ResourceActions;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.util.GetterUtil;
import com.liferay.portal.kernel.workflow.BaseWorkflowHandler;
import com.liferay.portal.kernel.workflow.WorkflowConstants;
import com.liferay.portal.kernel.workflow.WorkflowHandler;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocalService;

import java.io.Serializable;
import java.util.Locale;
import java.util.Map;
```

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

/**
 * Assignments workflow handler.
 *
 * @author liferay
 */
@Component(
    immediate = true,
    service = WorkflowHandler.class
)
public class AssignmentWorkflowHandler extends BaseWorkflowHandler<Assignment> {

    @Override
    public String getClassName() {

        return Assignment.class.getName();
    }

    @Override
    public String getType(Locale locale) {

        return _resourceActions.getModelResource(locale, getClass
Name());
    }

    @Override
    public Assignment updateStatus(
        int status, Map<String, Serializable> workflowContext)
        throws PortalException {

        long userId = GetterUtil.getLong(
            (String) workflowContext.get(WorkflowConstants.CONTEX
T_USER_ID));

        long resourcePrimKey = GetterUtil.getLong(
            (String) workflowContext.get(

```

```

        WorkflowConstants.CONTEXT_ENTRY_CLASS_PK));

    ServiceContext serviceContext =
        (ServiceContext) workflowContext.get("serviceContext");
}

return _assignmentLocalService.updateStatus(
    userId, resourcePrimKey, status, serviceContext);
}

@Reference
private AssignmentLocalService _assignmentLocalService;

@Reference
private ResourceActions _resourceActions;

}

```



Implement Support for Status in the Getter Methods

Now our entities support workflows on the service layer but we also have to be able to query them by their status so that for example draft entities are not shown to unauthorized users.

For the sake of exercise, we'll add a new signature for `getAssignmentsByKeywords()` only, taking the `status` field into account.

- Open** the class

```
com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl
```

in the *gradebook-service* module.

- Add** new signatures for the `getAssignmentsByKeywords()` and `getAssignmentsCountByKeywords()` as follows:

```

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

```

```

        DynamicQuery assignmentQuery = getKeywordSearchDynamicQuery(g
roupId, keywords);

        if (status != WorkflowConstants.STATUS_ANY) {
            assignmentQuery.add(RestrictionsFactoryUtil.eq("status",
status));
        }

        return assignmentLocalService.dynamicQuery(
            assignmentQuery, start, end, orderByComparator);
    }

    public long getAssignmentsCountByKeywords(
        long groupId, String keywords, int status) {

        DynamicQuery assignmentQuery = getKeywordSearchDynamicQuery(g
roupId, keywords);

        if (status != WorkflowConstants.STATUS_ANY) {
            assignmentQuery.add(RestrictionsFactoryUtil.eq("status",
status));
        }

        return assignmentLocalService.dynamicQueryCount(assignmentQue
ry);
    }
}

```

These methods are called through the remote service, so let's add facades for them:

- Open** the class

```
com.liferay.training.gradebook.service.impl.AssignmentServiceImp
1.java in the gradebook-service module.
```

- Implement** the new facade methods as follows:

```

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

```

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

    public long getAssignmentsCountByKeywords(
        long groupId, String keywords, int status) {

        return assignmentLocalService.getAssignmentsCountByKeywords(
            groupId, keywords, status);
    }
}
```

3. **Rebuild** the service.

Implement Workflow Support for Search

Implement a pre filter for Assignments search so that only approved entities are shown:

1. **Create** a class

```
com.liferay.training.gradebook.internal.search.spi.model.query.c
ontributor.AssignmentModelPreFilterContributor 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.SearchContext;
import com.liferay.portal.kernel.search.filter.BooleanFilter;
import com.liferay.portal.search.spi.model.query.contributor.Mode
lPreFilterContributor;
import com.liferay.portal.search.spi.model.registrar.ModelSearchS
ettings;

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 = ModelPreFilterContributor.class
)
public class AssignmentModelPreFilterContributor
    implements ModelPreFilterContributor {

    @Override
    public void contribute(
        BooleanFilter booleanFilter, ModelSearchSettings modelSea
rchSettings,
        SearchContext searchContext) {

        addWorkflowStatusFilter(
            booleanFilter, modelSearchSettings, searchContext);
    }

    protected void addWorkflowStatusFilter(
        BooleanFilter booleanFilter, ModelSearchSettings modelSea
rchSettings,
        SearchContext searchContext) {

        workflowStatusModelPreFilterContributor.contribute(
            booleanFilter, modelSearchSettings, searchContext);
    }

    @Reference(target = "(model.pre.filter.contributor.id=Workflo
wStatus)")
    protected ModelPreFilterContributor workflowStatusModelPreFil
terContributor;

}
```

Implement Support for Workflow Status in the MVC Render Command

Next we'll change the implementation of the `ViewAssignmentsMVCRenderCommand`, which is responsible for listing the Assignments, so that it takes the `status` field into account and for the sake of exercise, exposes drafted entities only for the administrators:

1. **Open** the class

```
com.liferay.training.gradebook.web.portlet.action.ViewAssignment  
sMVCRenderCommand
```

2. **Replace** the contents of the class with the following. See the highlighted lines for changes:

```
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.MVCRenderCom  
mand;  
import com.liferay.portal.kernel.security.permission.PermissionCh  
ecker;  
import com.liferay.portal.kernel.theme.ThemeDisplay;  
import com.liferay.portal.kernel.util.OrderByComparator;  
import com.liferay.portal.kernel.util.OrderByComparatorFactoryUti  
l;  
import com.liferay.portal.kernel.util.ParamUtil;  
import com.liferay.portal.kernel.util.Portal;  
import com.liferay.portal.kernel.util.WebKeys;  
import com.liferay.portal.kernel.workflow.WorkflowConstants;  
import com.liferay.training.gradebook.model.Assignment;  
import com.liferay.training.gradebook.service.AssignmentService;  
import com.liferay.training.gradebook.web.constants.GradebookPort  
letKeys;  
import com.liferay.training.gradebook.web.constants.MVCCommandNam  
es;  
import com.liferay.training.gradebook.web.display.context.Assignm  
entsManagementToolbarDisplayContext;  
import com.liferay.training.gradebook.web.internal.security.permit
```

```
ssion.resource.AssignmentPermission;

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=" + MVCCCommandNames.VIEW_ASSIGNMENTS
    },
    service = MVCRenderCommand.class
)
public class ViewAssignmentsMVCRenderCommand implements MVCRender
Command {

    @Override
    public String render(
        RenderRequest renderRequest, RenderResponse renderRespons
e)
        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 assignment 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");

// Get the workflow status for the list.

int status = getAllowedWorkflowStatus(renderRequest);

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

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

// Set request attributes.

renderRequest.setAttribute("assignments", assignments);
```

```
        renderRequest.setAttribute(
            "assignmentCount", _assignmentService.getAssignmentsCountByKeywords(
                themeDisplay.getScopeGroupId(), keywords, status)
        );

    }

    /**
     * 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 assignmentsManagementToolbarDisplayContext =
            new AssignmentsManagementToolbarDisplayContext(
                liferayPortletRequest, liferayPortletResponse,
                _portal.getHttpServletRequest(renderRequest));

        renderRequest.setAttribute(
            "assignmentsManagementToolbarDisplayContext",
            assignmentsManagementToolbarDisplayContext);
    }

}

/**
 * Returns workflow status current user is allowed to see.
*
```

```
        * This simple example returns ANY status for company admin  
        * and  
        * APPROVED for other users.  
        *  
        * @param renderRequest  
        * @return  
        */  
  
    private int getAllowedWorkflowStatus(RenderRequest renderRequest) {  
  
        ThemeDisplay themeDisplay =  
            (ThemeDisplay) renderRequest.getAttribute(WebKeys.THEME_DISPLAY);  
  
        PermissionChecker permissionChecker = themeDisplay.getPermissionChecker();  
  
        int status;  
  
        if (permissionChecker.isCompanyAdmin()) {  
            status = WorkflowConstants.STATUS_ANY;  
        } else {  
            status = WorkflowConstants.STATUS_APPROVED;  
        }  
  
        return status;  
    }  
  
    @Reference  
    protected AssignmentPermission _assignmentPermission;  
  
    @Reference  
    protected AssignmentService _assignmentService;  
  
    @Reference  
    private Portal _portal;  
}
```

Implement Support for Workflow Status in the JSP Files

The last thing to do is to show status in the user interface. Let's add the `status` column to the JSP file responsible for showing the Assignments list columns.

1. **Open** the file `/src/main/resources/META-INF/resources/assignment/entry_search_column.jspf` in the *gradebook-web* module.
2. **Replicate** the file contents with the following (see highlighted code for changes):

```

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

<portlet:renderURL var="viewAssignmentURL">
    <portlet:param name="mvcRenderCommandName" value="<%="MVCCommandNames.VIEW_ASSIGNMENT %>" />
    <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().equals("descriptive")}'>

        <%-- 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(),
true);
%>

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

<h4>
    <aui:a href="${viewAssignmentURL}">
        ${entry.getTitle(locale)}
    </aui:a>
</h4>

<h5 class="text-default">
    <aui:workflow-status
        markupView="lexicon"
        showIcon="<%= true %>"
        showLabel="<%= false %>"
        status="${entry.status}"
    />
</h5>

</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().equals("icon")}'>
    <%

```

```
        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.getTitle(locale)}"
    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>

<%-- Workflow status --%>

<aui:workflow-status
    markupView="lexicon"
    showIcon="<%= false %>"
    showLabel="<%= false %>"
    status="${entry.status}"
/>

<div class="truncate-text">

<%-- Strip HTML --%>

<%=HtmlUtil.stripHtml(entry.getDescription(locale)) %>
</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(locale) %>"
    />

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

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

    <%-- Workflow status --%>

    <liferay-ui:search-container-column-status
        name="status"
    />

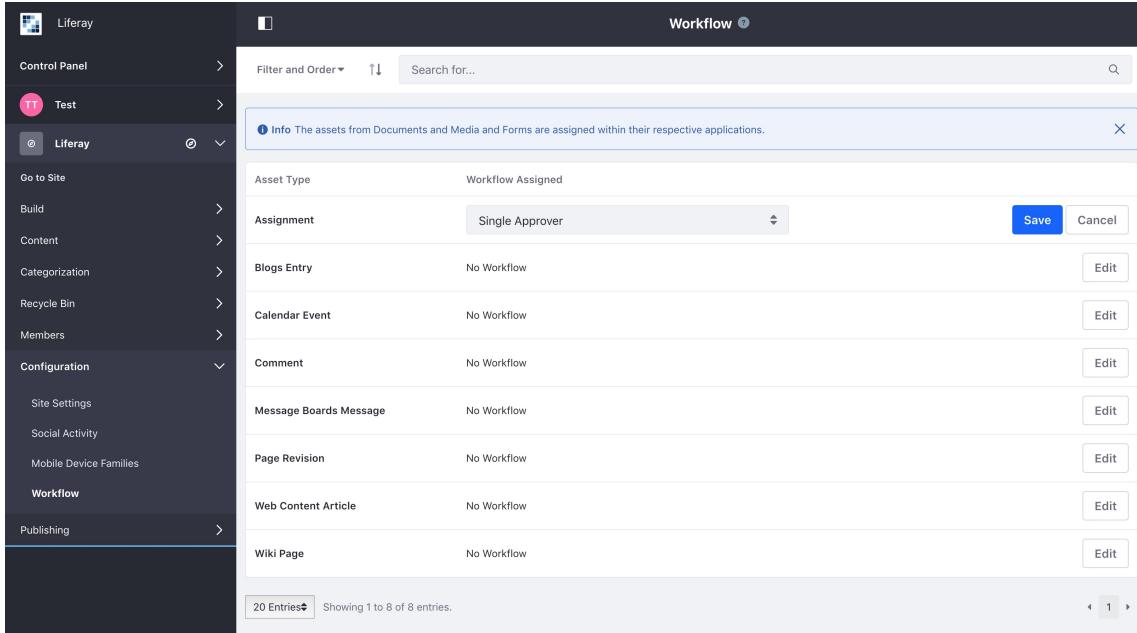
    <liferay-ui:search-container-column-jsp
        name="actions"
        path="/assignment/entry_actions.jsp"
    />
</c:otherwise>
</c:choose>
```



Test the Application

To be able to test, we have to enable and define a workflow for Assignments:

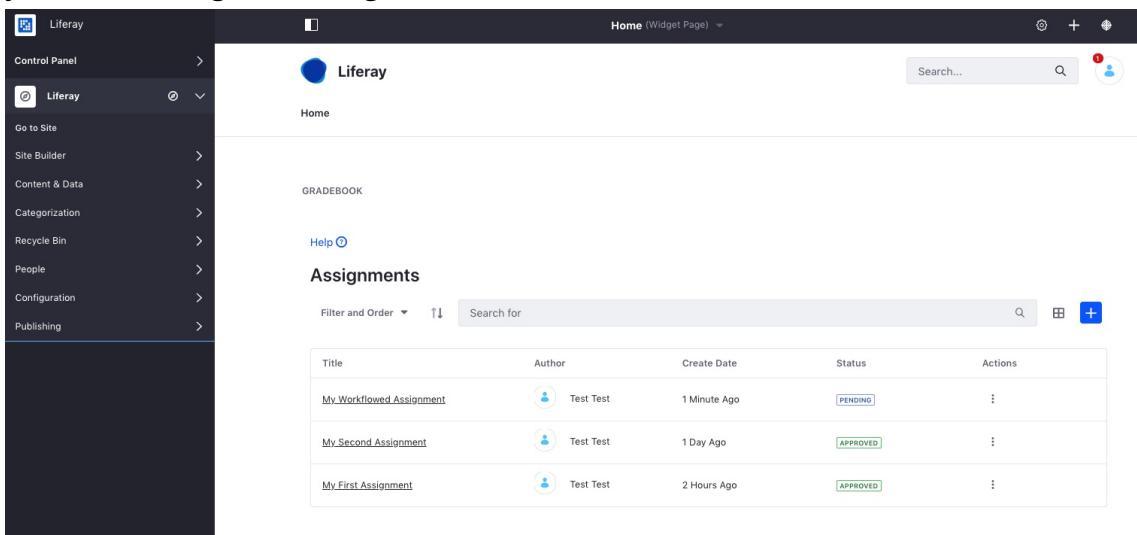
1. **Open** your browser and sign in if necessary.
2. **Go to Site Administration → Configuration → Workflow.**
3. **Set Assignment workflow to Single Approver.**



The screenshot shows the Liferay Control Panel with the 'Workflow' configuration page open. The left sidebar shows 'Workflow' selected under 'Configuration'. The main panel lists various asset types and their assigned workflows. The 'Assignment' asset type is highlighted with a dropdown menu showing 'Single Approver' selected. Other asset types like 'Blogs Entry', 'Calendar Event', 'Comment', etc., are listed with 'No Workflow' assigned. There are 'Edit' buttons next to each row.

Asset Type	Workflow Assigned
Assignment	Single Approver
Blogs Entry	No Workflow
Calendar Event	No Workflow
Comment	No Workflow
Message Boards Message	No Workflow
Page Revision	No Workflow
Web Content Article	No Workflow
Wiki Page	No Workflow

4. **Open** the Gradebook application and create an Assignment. The status on the list should now be pending.
5. **Create** a new assignment. After refreshing the page, you should see a notification on your avatar image indicating a new workflow event:



The screenshot shows the Liferay Control Panel with the 'Gradebook' application open. The left sidebar shows 'Gradebook' selected. The main panel displays a table of assignments with columns for Title, Author, Create Date, Status, and Actions. Three assignments are listed: 'My Workflowed Assignment' (Status: PENDING), 'My Second Assignment' (Status: APPROVED), and 'My First Assignment' (Status: APPROVED). A notification badge (with the number 1) is visible on the user's profile icon in the top right corner.

Title	Author	Create Date	Status	Actions
My Workflowed Assignment	Test Test	1 Minute Ago	PENDING	⋮
My Second Assignment	Test Test	1 Day Ago	APPROVED	⋮
My First Assignment	Test Test	2 Hours Ago	APPROVED	⋮

Now you can manage the workflows for Assignments as for any other Liferay assets.

Optional Exercise

Publish a REST Service

Exercise Goals

- Create a Liferay Module project using the Rest template
- Declare dependencies
- Implement the `AssignmentRestApplication` class
- Final code review
- Deploy and test
- Create an OAuth 2.0 application
- Test

In this exercise, we will be adding a REST API to the service layer by way of a JAX-RS Whiteboard. We will be using the standard JAX-RS Whiteboard annotations. You can find more information regarding JAX-RS Whiteboard in the official [OSGi documentation](#). We'll expose two REST methods: one to get all the Assignments and another one to look up a specific Assignment by its ID.

a cURL command line client is needed for this exercise. Windows users can download a client for example [here](#).

Create a Liferay Module Project

Option 1: Use the Command Line Blade tools

1. **Open** command line shell in your Liferay Workspace `modules` folder.
2. **Run** command:

```
blade create -t rest -v 7.2 -p com.liferay.training.gradebook.  
rest -c AssignmentRest gradebook-rest
```

3. **Run** Gradle refresh on the IDE.

Option 2: Use Developer Studio Wizard

1. **Launch** the *Liferay Module Project* wizard in Developer Studio.
2. **Use** the following information for the first step:
 - **Project Name:** "indexer-post-processor"
 - **Build Type:** Gradle
 - **Liferay Version:** 7.2
 - **Project Template:** rest
3. **Click Next** and use the following information in the second step:
 - **Component Class Name:** "AssignmentRest"
 - **Package Name:** "com.liferay.training.gradebook.rest"
4. **Click Finish** to close the wizard

Resolve Dependencies

We'll need to resolve dependencies for the portal kernel, servlet, portlet and Gradebook API:

1. **Open** the `build.gradle`.
2. **Add** the following new dependencies:

```
compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"  
compileOnly group: "javax.portlet", name: "portlet-api"  
compileOnly group: "javax.servlet", name: "javax.servlet-api"  
compileOnly project(":modules:gradebook:gradebook-api")
```

Implement the GradebookRestApplication Class

Well implement a simple rest application which you can use to fetch the list of all the Assignment or just a single one by its ID. OAuth will be disabled in this application.

1. **Open** the `AssignmentRestApplication` class, generated by the project template.
2. **Replace** the class contents with:

```
package com.liferay.training.gradebook.rest.application;
```

```
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.json.JSONFactoryUtil;
import com.liferay.portal.kernel.model.Company;
import com.liferay.portal.kernel.model.Group;
import com.liferay.portal.kernel.service.CompanyService;
import com.liferay.portal.kernel.service.GroupService;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Set;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Application;
import javax.ws.rs.core.MediaType;

import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Reference;
import org.osgi.service.jaxrs.whiteboard.JaxrsWhiteboardConstants
;

/**
 * Simple REST application.
 *
 * @author liferay
 */
@Component(
    property = {
        JaxrsWhiteboardConstants.JAX_RS_APPLICATION_BASE + "=/gradebook-rest",
        JaxrsWhiteboardConstants.JAX_RS_NAME + "=Gradebook.Rest",
        "liferay.auth.verifier=true", // default
        "liferay.oauth2=true" // default
    },

```

```
        service = Application.class
    )
public class AssignmentRestApplication extends Application {

    public Set<Object> getSingletons() {
        return Collections.<Object>singleton(this);
    }

    @GET
    @Path("/assignments")
    @Produces({
        MediaType.APPLICATION_JSON
    })
    public String getAssignments() {

        try {
            List<Assignment> assignments = new ArrayList<Assignment>();

            Company company =
                _companyService.getCompanyById(_portal.getDefault
CompanyId());

            List<Group> groups =
                _groupService.getGroups(company.getCompanyId(), 0
, true);

            for (Group group : groups) {
                assignments.addAll(
                    _assignmentService.getAssignmentsByGroupId(
                        group.getGroupId()));
            }

            return JSONFactoryUtil.serialize(assignments);
        }
        catch (PortalException pe) {
            pe.printStackTrace();
            return "[{}]";
        }
    }
}
```

```
}

@GET
@Path("/assignment/{assignmentid}")
@Produces({
    MediaType.APPLICATION_JSON
})
public String getAssignment(
    @PathParam("assignmentid") long assignmentId) {

    try {
        return JSONFactoryUtil.serialize(
            _assignmentService.getAssignment(assignmentId));
    }
    catch (Exception e) {
        e.printStackTrace();
        return "{}";
    }
}

@Reference
private AssignmentService _assignmentService;

@Reference
private CompanyService _companyService;

@Reference
private GroupService _groupService;

@Reference
private Portal _portal;
}
```

Final Code Review

The complete implementation files will look like this:

build.gradle

```
dependencies {
```

```
    compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"
    compileOnly group: "javax.portlet", name: "portlet-api"
    compileOnly group: "javax.servlet", name: "javax.servlet-api"
    compileOnly group: "javax.ws.rs", name: "javax.ws.rs-api"
    compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"
    compileOnly group: "org.osgi", name: "org.osgi.service.jaxrs"
    compileOnly project(":modules:gradebook:gradebook-api")
}
```

AssignmentRestApplication.java

```
package com.liferay.training.gradebook.rest.application;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.json.JSONFactoryUtil;
import com.liferay.portal.kernel.model.Company;
import com.liferay.portal.kernel.model.Group;
import com.liferay.portal.kernel.service.CompanyService;
import com.liferay.portal.kernel.service.GroupService;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Set;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Application;
import javax.ws.rs.core.MediaType;

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

```
import org.osgi.service.jaxrs.whiteboard.JaxrsWhiteboardConstants
;

/**
 * Simple REST application.
 *
 * @author liferay
 */
@Component(
    property = {
        JaxrsWhiteboardConstants.JAX_RS_APPLICATION_BASE + "=/gra
debook-rest",
        JaxrsWhiteboardConstants.JAX_RS_NAME + "=Gradebook.Rest"
    },
    service = Application.class
)
public class AssignmentRestApplication extends Application {

    public Set<Object> getSingletons() {
        return Collections.<Object>singleton(this);
    }

    @GET
    @Path("/assignments")
    @Produces({
        MediaType.APPLICATION_JSON
    })
    public String getAssignments() {

        try {
            List<Assignment> assignments = new ArrayList<Assignme
nt>();

            Company company =
                _companyService.getCompanyById(_portal.getDefault
CompanyId());
            List<Group> groups =
                _groupService.getGroups(company.getCompanyId(), 0
, true);
        }
    }
}
```

```
        for (Group group : groups) {
            assignments.addAll(
                _assignmentService.getAssignmentsByGroupId(
                    group.getGroupId()));
        }

        return JSONFactoryUtil.serialize(assignments);
    }

    catch (PortalException pe) {
        pe.printStackTrace();
        return "[{}]";
    }
}

@GET
@Path("/assignment/{assignmentid}")
@Produces({
    MediaType.APPLICATION_JSON
})
public String getAssignment(
    @PathParam("assignmentid") long assignmentId) {

    try {
        return JSONFactoryUtil.serialize(
            _assignmentService.getAssignment(assignmentId));
    }
    catch (Exception e) {
        e.printStackTrace();
        return "{}";
    }
}

@Reference
private AssignmentService _assignmentService;

@Reference
private CompanyService _companyService;
```

```
    @Reference  
    private GroupService _groupService;  
  
    @Reference  
    private Portal _portal;  
}
```

Deploy and Test

1. **Open** your browser to <http://localhost:8080> and sign in.
2. **Open** your browser in <http://localhost:8080/o/gradebook-rest/assignments>. You'll get an access denied message:

```
<Forbidden>  
  <message>  
    Access denied to com.liferay.training.gradebook.rest.  
    application.AssignmentRestApplication#getAssignments  
  </message>  
</Forbidden>
```

The JAX-RS REST application requires OAuth 2.0 authorization by default and we have to configure that to grant access to the service.

Create an OAuth 2.0 Application

1. **Go to** the *Control Panel* → *Configuration* → *OAuth2 Administration*
2. **Click** the Add icon to add an application.
3. **Use** the following information for the first step:
 - **Application Name:** "Gradebook REST"
 - **Client Profile:** "Headless Server"
4. **Leave** defaults for the other values and click *Save*
5. **Use** a text editor to copy the following values on the next dialog:
 - **Client ID**
 - **Client Secret** (Click the *Edit* button to show the secret)
6. **Click** on the *Scopes* tab.
7. **Open** the *Gradebook.Rest* and check *read data on your behalf*
8. **Click** *Save*.

Test

1. **Open** the command line shell
2. **Use** a cURL request to get an access token. Use the client ID and secret values from the previous step :

```
curl http://localhost:8080/o/oauth2/token -d 'grant_type=client_credentials&client_id=[CLIENT_ID]&client_secret=[CLIENT_SECRET']'
```

You'll get the access token as a response:

```
{"access_token":"262e6fffb6faffef928b251efa9f15644526023d63a7a9a3d6527b94ef12b2c","token_type":"Bearer","expires_in":600,"scope":"Gradebook.Rest.everything.read"}
```

3. **Now** use the access token and call the Assignment REST service to list all the assignments:

```
curl -H 'Accept: application/json' -H "Authorization: Bearer [ACCESS_TOKEN]" http://localhost:8080/o/gradebook-rest/assignments
```

You'll get Assignments list as a response:

```
{"javaClass":"java.util.ArrayList","list":[{"contextName":"com.liferay.training.gradebook.service_1.0.0","javaClass":"com.liferay.training.gradebook.model.impl.AssignmentImpl","serializable":{"statusDate":null,"originalUuid":"ee4172af-a9d8-83cd-5f2a-cdd341813732","columnBitmask":0,"cachedModel":false,"groupId":20123,"dueDate":{"javaClass":"java.sql.Timestamp","time":1559260800000}, "description": "<?xml version='1.0' encoding='UTF-8'?><root available-locales=\"en_US\" default-locale=\"en_US\"><Description language-id=\"en_US\">Please write a short 1000 character essay about the concept of positivism ..."}, "groupOrder":1,"id":1,"modifiedDate":1559260800000,"name": "Assignment 1","status":1,"title": "Assignment 1"}]}
```

4. **Locate** any `assignmentId` value in the response and get a single assignment from our REST service:

```
curl -H 'Accept: application/json' -H "Authorization: Bearer [ACCESS_TOKEN]" http://localhost:8080/o/gradebook-rest/assignment/[ASSIGNMENT_ID]
```

Takeaways

You've seen in this exercise how easy it is to create a REST API for your Liferay Services. Because JAX-RS Whiteboard is a standard OSGi specification, you now have another convenient tool at your disposal to create APIs that others can use to easily integrate with your applications.

See Developer Network article (https://dev.liferay.com/en/develop/tutorials/-/knowledge_base/7-2/jax-rs-and-jax-ws) for more information about using JAX RS and JAX WS in your projects.

Optional Exercise

Implement Gradebook Logging

Exercise Goals

- Add the SLF4J API dependency to the service module's `build.gradle`
- Add logging code to the Gradebook service
- Create and configure with `module-log4j.xml`

Add the SLF4J API dependency to the Service Module

1. Open the `build.gradle` of the *gradebook-service* module.
2. Add the dependency for SLF4J in the `dependencies` block:

```
compileOnly group: "org.slf4j", name: "slf4j-api"
```

Add Logging Code to the Gradebook Service

1. Open the `com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl` class in the *gradebook-service* module.
2. Add logger variable instantiation to the very end of the class:

```
private static final Logger _logger = LoggerFactory.getLogger(AssignmentLocalServiceImpl.class);
```

3. Implement logging code to the `addAssignment()` method:

```
if (_logger.isInfoEnabled()) {  
    _logger.info("User " + userId + " is adding an assignment  
.");  
}
```

4. **Organize** imports and save the class. Changes should be deployed automatically.

If you test adding an assignment, you won't see any messages in the log yet because the logging level for the package `com.liferay` is set by default to `ERROR`. Next, we will add a Log4J configuration file and set the logging level for debugging purposes to `DEBUG`.

Create and Configure Logging with `module-log4j.xml`

1. **Create** a logging configuration file `src/main/resources/META-INF/module-log4j.xml` in the *gradebook-service* module.
2. **Implement** the file as follows:

```
<?xml version="1.0"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <!-- Set logging level to info -->

    <category name="com.liferay.training.gradebook.service.impl">
        <priority value="INFO" />
    </category>
</log4j:configuration>
```

3. **Create** a new Assignment in the Gradebook application. You should see a similar entry in your log:

```
2018-09-24 09:18:50.880 INFO [http-nio-8080-exec-2][AssignmentLocalServiceImpl:148] User 20139 is adding an assignment.
```

Optional Exercise

Implement Integration Tests

Exercise Goals

- Create a Gradebook test module
- Declare dependencies
- Configure test settings
- Implement the test class
- Final code review
- Run tests

Create a Gradebook Test Module

Option 1: Use the Command Line Blade tools

1. **Open** command line shell in your Liferay Workspace `modules` folder.
2. **Run** command:

```
blade create -t activator -v 7.1 gradebook-test
```

3. **Run** Gradle refresh on the IDE.

Option 2: Use Developer Studio Wizard

1. **Launch** the *Liferay Module Project* wizard in Developer Studio.
2. **Use** the following information for the first step:
 - **Project Name:** "gradebook-test"
 - **Build Type:** Gradle
 - **Liferay Version:** 7.2
 - **Project Template:** activator
3. **Click Finish** to close the wizard.

Clean the unnecessary resources created by the wizard and create a new source folder for integration tests:

1. **Delete** the `src/main` folder.
2. **Create** a new **source folder** `src/testIntegration` (watch the case).

Declare Dependencies

1. **Open** the `build.gradle`.
2. **Replace** the dependencies section with the following:

```
testIntegrationCompile group: "com.liferay.portal", name: "com.liferay.portal.kernel"
testIntegrationCompile group: "com.liferay", name: "com.liferay.arquillian.extension.junit.bridge", version: "1.0.11"
testIntegrationCompile group: "com.liferay.portal", name: "com.liferay.portal.test", version: "5.3.2"
testIntegrationCompile group: "com.liferay.portal", name: "com.liferay.portal.test.integration", version: "5.1.7"
testIntegrationCompile group: "com.liferay", name: "com.liferay.osgi.util"

testIntegrationCompile group: "javax.portlet", name: "portlet-api"

testIntegrationCompile group: "javax.servlet", name: "javax.servlet-api"
testIntegrationCompile group: "org.springframework", name: "spring-core", version: "4.3.3.RELEASE"

testIntegrationCompile group: "com.liferay", name: "com.liferay.property.string"
testIntegrationCompile group: "com.liferay", name: "com.liferay.property.lang"
testIntegrationRuntime group: "com.liferay", name: "com.liferay.property.concurrent"
testIntegrationRuntime group: "com.liferay", name: "com.liferay.property.memory"

testIntegrationCompile project(":modules:gradebook:gradebook-api")
)
```



Configure Test Settings

Configure the Gradle settings for the test Tomcat bundle and error logging:

1. Open the `build.gradle`.
2. Add the following configuration after the dependencies:

```
testIntegration {  
    ignoreFailures = false  
  
    testLogging {  
        events "started", "passed", "skipped", "failed", "standardOut", "standardError"  
        showExceptions true  
        exceptionFormat "full"  
        showCauses true  
        showStackTraces true  
    }  
}  
  
// Cleans the test bundle on startup.  
  
setUpTestableTomcat {  
    enabled = true  
}  
  
startTestableTomcat {  
    enabled = true  
}  
  
stopTestableTomcat {  
    enabled = true  
}
```

Implement the Test Class

1. Create a new Class

```
com.liferay.gradebook.service.impl.test.AssignmentLocalServiceImplTest.java
```

in the `testIntegration` source folder.

2. Annotate the class with `@RunWith` to make it an Arquillian test:

```
@RunWith(Arquillian.class)
public class AssignmentLocalServiceImplTest
```

3. Declare the Liferay JUnit class rule variable :

```
@ClassRule
@Rule
public static final AggregateTestRule aggregateTestRule =
    new LiferayIntegrationTestRule();
```

4. Add the required references to the end of the class:

```
@Inject
private AssignmentLocalService _assignmentLocalService;

@Inject
private GroupLocalService _groupLocalService;

@Inject
private UserLocalService _userLocalService;

@Inject
private Portal _portal;
```

5. Implement the test methods as follows:

```
@Test
public void testAddAssignment() throws Exception {
    int entriesCount = _assignmentLocalService.getAssignmentsCount();
```

```
ServiceContext serviceContext = new ServiceContext();
serviceContext.setUserId(getDefaultUserId());

Map<Locale, String>titleMap = new HashMap<Locale, String>();
titleMap.put(new Locale("en", "US"), "My Test Assignment Title");

Map<Locale, String>descriptionMap = new HashMap<Locale, String>();
titleMap.put(new Locale("en", "US"), "My Description");

Date dueDate = new Date();

_assignmentLocalService.addAssignment(getGuestGroupId(), titleMap, descriptionMap, dueDate, serviceContext);

assertEquals(++entriesCount, _assignmentLocalService.getAssignmentsCount());
}

@Test
public void testDeleteAssignment()
throws Exception {

int entriesCount = _assignmentLocalService.getAssignmentsCount();

ServiceContext serviceContext = new ServiceContext();
serviceContext.setUserId(getDefaultUserId());

Map<Locale, String>titleMap = new HashMap<Locale, String>();
titleMap.put(new Locale("en", "US"), "My Test Assignment Title");

Map<Locale, String>descriptionMap = new HashMap<Locale, String>();
titleMap.put(new Locale("en", "US"), "My Description");

Date dueDate = new Date();
```

```

        Assignment assignment = _assignmentLocalService.addAssignment
(getGuestGroupId(), titleMap, descriptionMap, dueDate, serviceCon
text);

        assertEquals(++entriesCount, _assignmentLocalService.getAssig
nmentsCount());

        _assignmentLocalService.deleteAssignment(assignment);

        assertEquals(--entriesCount, _assignmentLocalService.getAssig
nmentsCount());
    }

private long getDefaultUserId()
throws PortalException {

    return UserLocalServiceUtil.getDefaultUserId(
        _portal.getDefaultCompanyId());
}

private long getGuestGroupId()
throws PortalException {

    String groupName = GroupConstants.GUEST;

    return _groupLocalService.getGroup(
        _portal.getDefaultCompanyId(), groupName).getGroupId();
}

```

Final Code Review

The complete files will look like this:

build.gradle

```

dependencies {
    testIntegrationCompile group: "com.liferay.portal", name: "co
m.liferay.portal.kernel"
    testIntegrationCompile group: "com.liferay", name: "com.lifer

```

```
ay.arquillian.extension.junit.bridge", version: "1.0.11"
    testIntegrationCompile group: "com.liferay.portal", name: "co
m.liferay.portal.test", version: "5.3.2"
    testIntegrationCompile group: "com.liferay.portal", name: "co
m.liferay.portal.test.integration", version: "5.1.7"
    testIntegrationCompile group: "com.liferay", name: "com.liferay
osgi.util"

    testIntegrationCompile group: "javax.portlet", name: "portlet
-api"
    testIntegrationCompile group: "javax.servlet", name: "javax.s
ervlet-api"
    testIntegrationCompile group: "org.springframework", name: "s
pring-core", version: "4.3.3.RELEASE"

    testIntegrationCompile group: "com.liferay", name: "com.liferay
petra.string"
    testIntegrationCompile group: "com.liferay", name: "com.liferay
petra.lang"
    testIntegrationRuntime group: "com.liferay", name: "com.liferay
petra.concurrent"
    testIntegrationRuntime group: "com.liferay", name: "com.liferay
petra.memory"

    testIntegrationCompile project(":modules:gradebook:gradebook-
api")
}

testIntegration {
    ignoreFailures = false

    testLogging {
        events "started", "passed", "skipped", "failed", "standar
dOut", "standardError"
        showExceptions true
        exceptionFormat "full"
        showCauses true
        showStackTraces true
    }
}
```

```
// Cleans the test bundle on startup.

setUpTestableTomcat {
    enabled = true
}

startTestableTomcat {
    enabled = true
}

stopTestableTomcat {
    enabled = true
}
```

AssignmentLocalServiceImplTest.java

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

import static org.junit.Assert.assertEquals;

import com.liferay.arquillian.extension.junit.bridge.junit.Arquillian;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.model.GroupConstants;
import com.liferay.portal.kernel.service.GroupLocalService;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.UserLocalService;
import com.liferay.portal.kernel.service.UserLocalServiceUtil;
import com.liferay.portal.kernel.test.rule.AggregateTestRule;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.portal.test.rule.Inject;
import com.liferay.portal.test.rule.LiferayIntegrationTestRule;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocalService;

import java.util.Date;
```

```
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

import org.junit.ClassRule;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

/**
 * @author liferay
 */
@RunWith(Arquillian.class)
public class AssignmentLocalServiceImplTest {

    @ClassRule
    @Rule
    public static final AggregateTestRule aggregateTestRule =
        new LiferayIntegrationTestRule();

    @Test
    public void testAddAssignment() throws Exception {

        int entriesCount = _assignmentLocalService.getAssignments
Count();

        ServiceContext serviceContext = new ServiceContext();
        serviceContext.setUserId(getDefaultUserId());

        Map<Locale, String>titleMap = new HashMap<Locale, String>
();
        titleMap.put(new Locale("en", "US"), "My Test Assignment
Title");

        Map<Locale, String>descriptionMap = new HashMap<Locale, S
tring>();
        titleMap.put(new Locale("en", "US"), "My Description");

        Date dueDate = new Date();
    }
}
```

```
        _assignmentLocalService.addAssignment(getGuestGroupId(),
titleMap, descriptionMap, dueDate, serviceContext);

        assertEquals(++entriesCount, _assignmentLocalService.getA
ssignmentsCount());
    }

    @Test
    public void testDeleteAssignment()
        throws Exception {

        int entriesCount = _assignmentLocalService.getAssignments
Count();

        ServiceContext serviceContext = new ServiceContext();
        serviceContext.setUserId(getDefaultUserId());

        Map<Locale, String> titleMap = new HashMap<Locale, String>
();
        titleMap.put(new Locale("en", "US"), "My Test Assignment
Title");

        Map<Locale, String> descriptionMap = new HashMap<Locale, S
tring>();
        titleMap.put(new Locale("en", "US"), "My Description");

        Date dueDate = new Date();

        Assignment assignment = _assignmentLocalService.addAssign
ment(getGuestGroupId(), titleMap, descriptionMap, dueDate, servic
eContext);

        assertEquals(++entriesCount, _assignmentLocalService.getA
ssignmentsCount());

        _assignmentLocalService.deleteAssignment(assignment);

        assertEquals(--entriesCount, _assignmentLocalService.getA
ssignmentsCount());
    }
}
```

```
private long getDefaultUserId()
throws PortalException {

    return UserLocalServiceUtil.getDefaultUserId(
        _portal.getDefaultCompanyId());
}

private long getGuestGroupId()
throws PortalException {

    String groupName = GroupConstants.GUEST;

    return _groupLocalService.getGroup(
        _portal.getDefaultCompanyId(), groupName).getGroupId();
}

@Inject
private AssignmentLocalService _assignmentLocalService;

@Inject
private GroupLocalService _groupLocalService;

@Inject
private UserLocalService _userLocalService;

@Inject
private Portal _portal;

}
```

Run the Test

1. Run the Gradle `testIntegration` task.

Watch the output on the console.

Optional Exercise

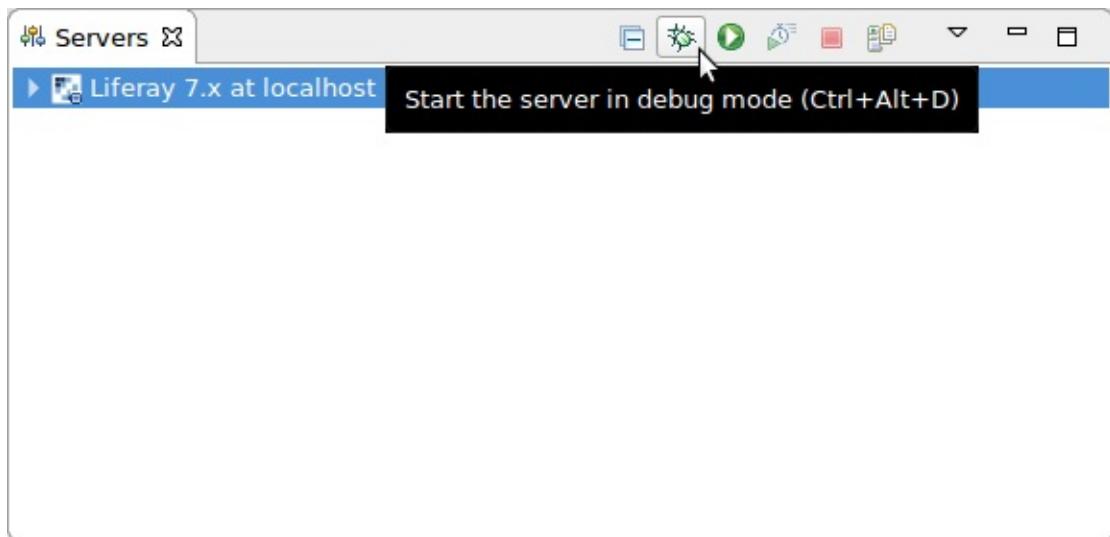
Debug the Gradebook

Exercise Goals

- (Re)start Tomcat in debug mode
- Add breakpoints
- Use debugger steps to control program execution

(Re)start Tomcat in Debug Mode

1. **Stop** the Liferay server.
2. **Start** the Liferay server in debug mode by clicking the "bug" icon:

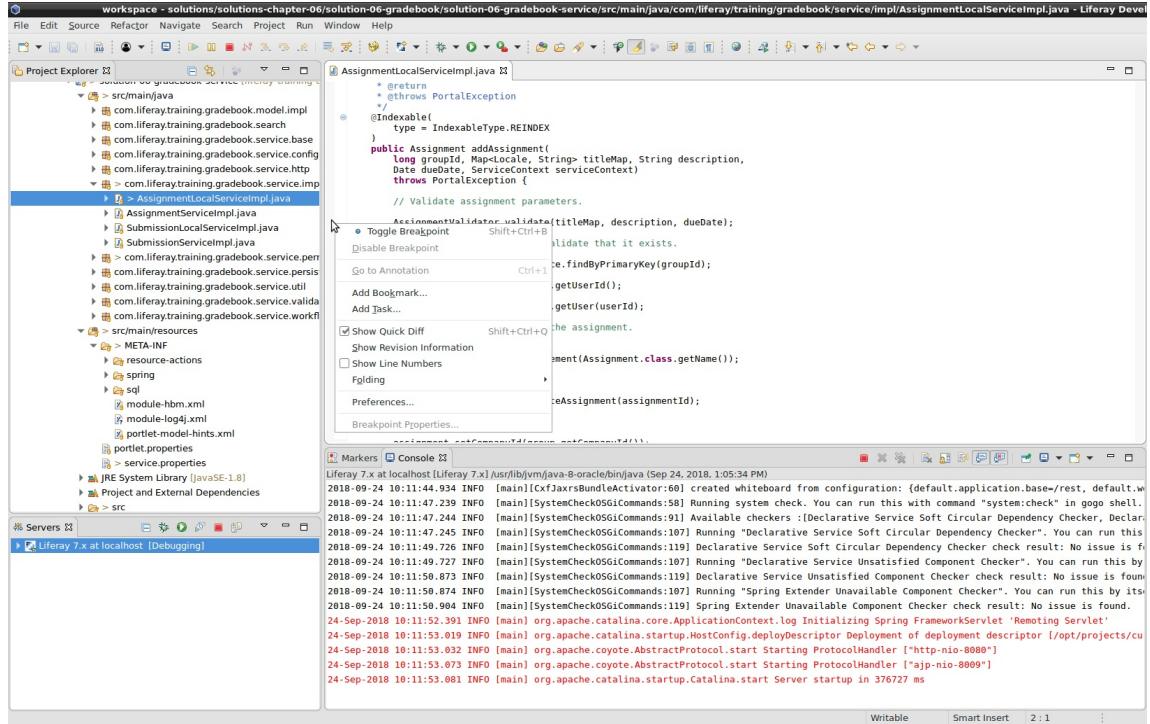


Add Breakpoint

1. **Open** `com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl` class in the *gradebook-service* module.
2. **Find** call to the `_assignmentValidator.validate()` in the beginning of

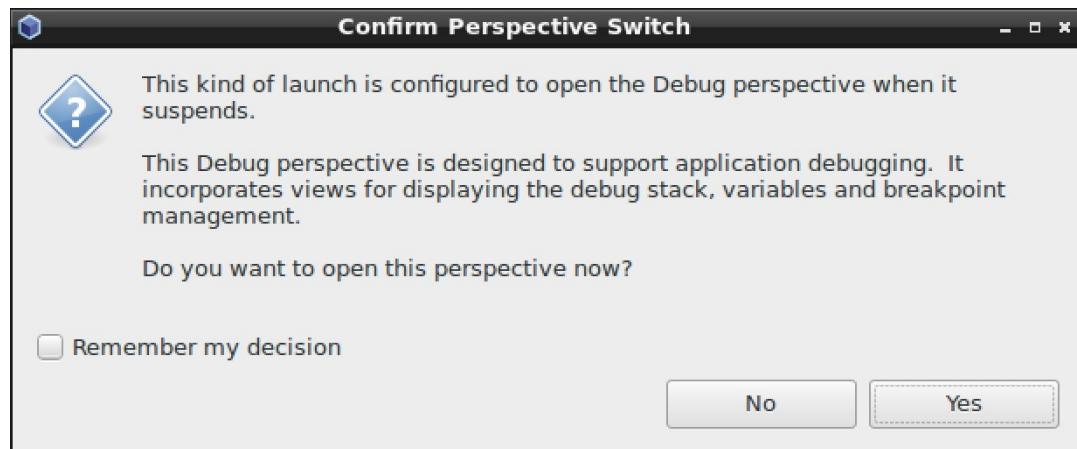
`addAssignment()` method.

- Right-click on the left side of the line (in the margin) and select *Toggle Breakpoint* from the context menu:



- Open your browser and add a new assignment in the Gradebook application.

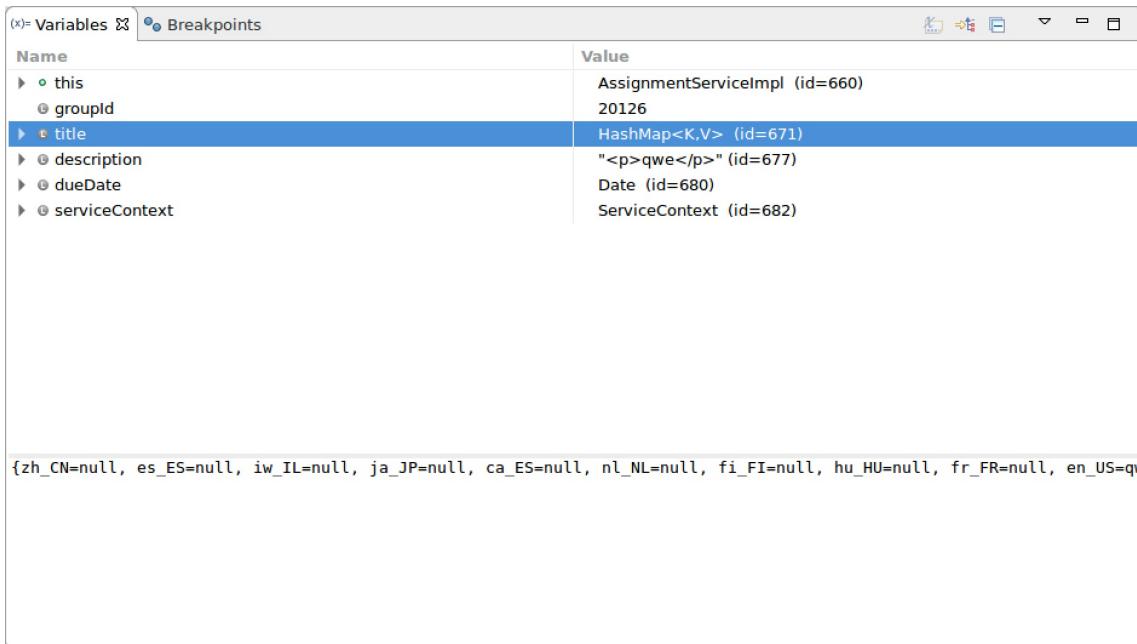
- Adding an assignment should pause in the breakpoint you just defined. In the IDE, there should be a dialog asking to switch to the debug perspective. Click "Yes".



- Click Yes to switch to the debug perspective.

- The application execution is now halted at the breakpoint. Let's check what the values are for our assignment title.

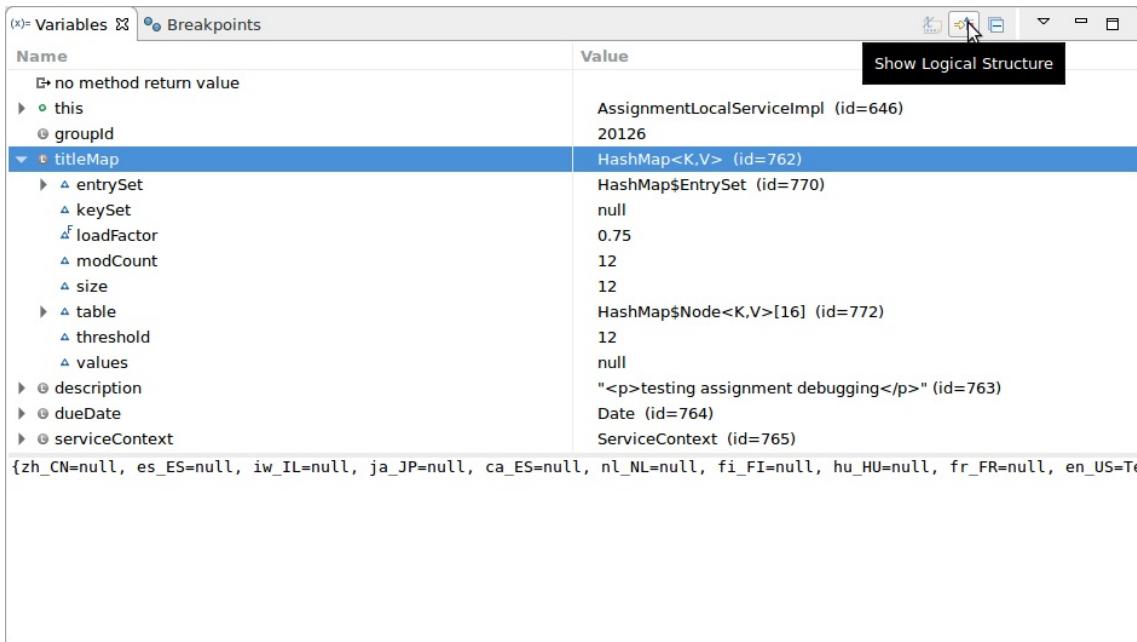
- Click the *Variables* to show the variables stack:



- From this panel, you can check and manipulate variable values on the fly.
- Click** the *titleMap* variable and browse it. In this case, you can see the Map values easily in the bottom panel, but most of the time it's hard to find some specific Array or Map value there, which is why we need the *Logical Structure view*.

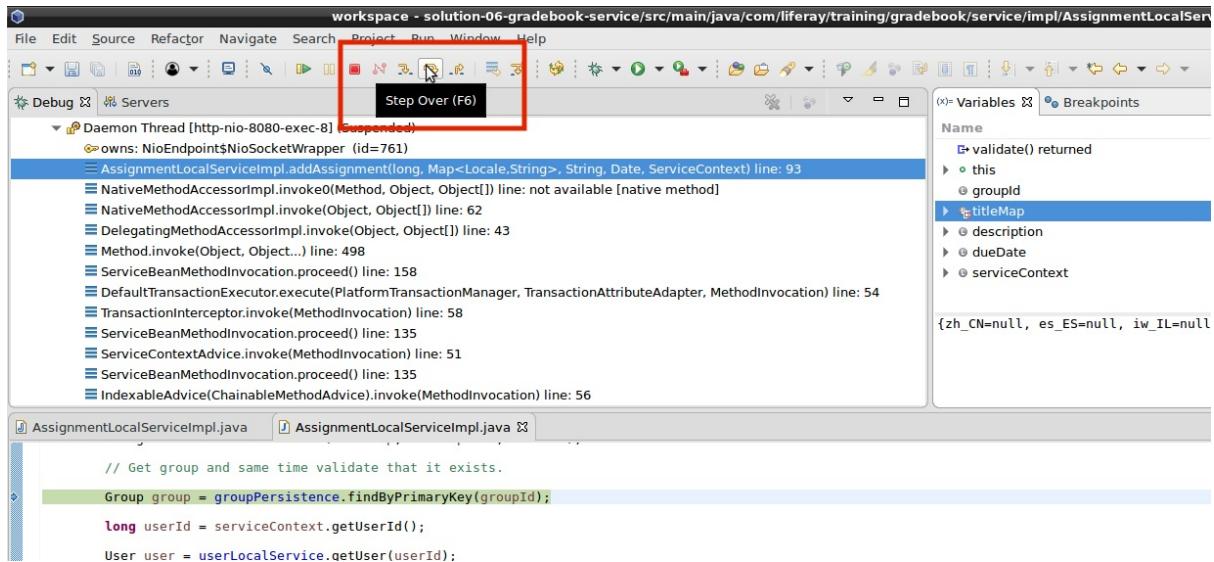
7. Click the *Show Logical Structure* button to show the Map's logical structure

8. Browse the *titleMap* variable again.



In the debug mode, you can fully control application execution. You can move back and forward as well as dive into the method calls. Try *Step over* to take one step forward in the method execution. Try *Step into* to go to the method behind the call. Watch the variables

panel while you move. Click *Resume* to continue application execution:



After you're done with the exercise, restart the server in normal mode.

Optional Exercise

Troubleshoot Module Deployment

Exercise Goals

- Deploy the exercise modules
- Troubleshoot the failing deployment

Deploy the Exercise Modules

1. **Copy** the two modules `failing-api` and `failing-service` from the provided `exercise-files/06-real-world-application/troubleshoot-module-deployment` folder to your `training-workspace/modules` folder.
2. **Run** Gradle refresh on the Liferay Workspace
3. **Deploy** the modules to the Liferay server

You'll notice that the modules won't start and you won't even get any log message.

Troubleshoot the Failing Deployment

1. **Login** to Gogo shell and list bundles using the `lb` command. You'll notice that the exercise modules `failing-api` and `failing-service` are in the `installed` state.

Remember that if a bundle can't go into the `resolved` state, there's a dependency problem.
2. **Use** the Gogo commands we learned in the *Module 4 - Managing OSGi Bundles* and find out what the two problems are. As you work with the code, any changes will be hot deployed automatically.

The exercise is completed when you get the following message in the log:

```
2019-04-08 11:12:54.374 INFO [Thread-2496][BundleStartStopLogger :42] STARTED com.liferay.training.deployment.failing.service_1.0.0 [979]
2019-04-08 11:12:54.374 INFO [Thread-2496][FailingService:36] ##
#####
2019-04-08 11:12:54.374 INFO [Thread-2496][FailingService:37] Task completed successfully!
2019-04-08 11:12:54.374 INFO [Thread-2496][FailingService:38] ##
#####
2019-04-08 11:12:54.703 INFO [Thread-2496][BundleStartStopLogger :39] STARTED com.liferay.training.deployment.failing.api_1.0.0 [979]
```

Hint: You might want to start with the Gogo shell tool for diagnosing dependency problems. You can ask your trainer any time for hints.