***Service Builder*** is Liferay’s code generation tool for defining object models and mapping those models to SQL databases. By defining your model in a single XML file, you can generate your object model (the M in MVC), your service layer, and your persistence layer all in one shot.

In the service.xml file, This defines the author, namespace, and the entity name. The namespace keeps the database field names from conflicting. The last tag is the opening tag for the Guestbook entity definition. In this tag, you enable local and remote services for the entity, define its name, and specify that it should have a [universally unique identifier (UUID)](https://en.wikipedia.org/wiki/Universally_unique_identifier).

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

* The groupId defines the ID of the Site in Liferay DXP where the entity instance exists. The companyId is the primary key of a [portal instance](https://help.liferay.com/hc/en-us/articles/360029131551-Introduction-to-Setting-Up).
* **Audit** fields:-

<!-- Audit fields -->

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

<column name="userName" type="String" />

<column name="createDate" type="Date" />

<column name="modifiedDate" type="Date" />

<!-- Status fields -->

<column name="status" type="int" />

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

<column name="statusByUserName" type="String" />

<column name="statusDate" type="Date" />

<finder name="GroupId" return-type="Collection">

<finder-column name="groupId" />

</finder>

* Service Builder is based on a design philosophy called loose coupling. It generates three layers of your application: the model, the service, and the persistence layers. Loose coupling means you can swap out the persistence layer with little to no change in the model and service layers. The model is in the -api module, and the service and persistence layers are in the -service module.
* Modules are the core building blocks of Liferay DXP applications. Every application is made from one or more modules. Modules are the collection of classes and packages with manifest file.Modules can be web modules or [OSGi](https://www.osgi.org/) modules.
* Each module is packaged as a JAR file that contains a manifest file. The manifest is needed for the container to recognize the module.

**Web apps in Liferay DXP** are called portlets. Like many web apps, portlets process requests and generate responses. In the response, the portlet returns content (e.g. HTML, XHTML) for display in browsers.

mvcPath is the path of .jsp file where you will render.

**Component :-** a component is the object that contains the core functionality. A Component is managed by a component framework or container. Components are deployed inside modules, and they’re created, started, stopped, and destroyed as needed by the container.

With Declarative Services, you declare that an object is a component, and you define data about the component so the container knows how to manage it.

@Component(

immediate = true,

property = {

"com.liferay.portlet.display-category=category.social",

"com.liferay.portlet.instanceable=false",

"com.liferay.portlet.scopeable=true",

"javax.portlet.display-name=Guestbook",

"javax.portlet.expiration-cache=0",

"javax.portlet.init-param.template-path=/",

"javax.portlet.init-param.view-template=/guestbook/view.jsp",

"javax.portlet.resource-bundle=content.Language",

"javax.portlet.security-role-ref=power-user,user",

"javax.portlet.supports.mime-type=text/html"

},

service = Portlet.class

)

**Portlet phase(2):**-

* Render Phase
* Action

1. The product menu is split into three sections: the Control Panel, the User menu, and the Sites menu.

**Portlet Mode:--**

1. View
2. Edit
3. Help

**Resource and permision:**

* **Define** all resources and permissions
* **Register** all defined resources in the permissions system
* **Associate** permissions with resources
* **Check** for permission before returning resources

1. There are therefore two kinds of permissions: portlet permissions and resource (or model) permissions. Portlet permissions protect access to global functions, such as *Add Entry.*
2. *In the /src/main/resources/META-INF folder, create a subfolder called resource-actions.*
3. *Create a new file in this folder called default.xml*
4. *In guestbook-service’s src/main/resources folder, create a file called portlet.properties.*
5. *In this file, place the following property:  
   resource.actions.configs=META-INF/resource-actions/default.xml*

***Enabling Search and Indexing:-***

* *Registration:*
  + *GuestbookSearchRegistrar registers the search services to the search framework for the Guestbook entity.*
* *Indexing:*
  + *GuestbookModelDocumentContributor controls which Guestbook fields are indexed in the search engine.*
  + *GuestbookModelIndexerWriterContributor configures the re-indexing and batch re-indexing behavior for Guestbooks.*
* *Querying:*
  + *GuestbookKeywordQueryContributor contributes clauses to the ongoing search query.*
  + *GuestbookModelPreFilterContributor controls how search results are filtered before they’re returned from the search engine.*
* *Generating Result Summaries:*
  + *GuestbookModelSummaryContributor constructs the result summary for Guestbooks, including specifying which fields to use.*

*The annotations @Activate and @Deactivate ensure each method is invoked as soon as the Component is started (activated) or when it’s about to be stopped (deactivated).*

***Log in jsp:-***

*<%! private static Log \_log = LogFactoryUtil.getLog("html.guestbook.view\_search\_jsp");*

*%>*

***The asset framework*** *transforms entities into a common format that can be published anywhere in your Site. Web content articles, blog posts, wiki articles, and documents are some asset-enabled entities that come out-of-the-box.*

*As background, the term asset* ***refers to any type of content in the portal****.*

***The asset framework includes these features:***

* *Tags and categories*
* *Comments and ratings*
* *Related assets (a.k.a. Asset links)*
* *Faceted search*
* *Integration with the Asset Publisher portlet*
* *Integration with the Search portlet*
* *Integration with the Tags Navigation, Tag Cloud, and Categories Navigation portlets*

***There are five steps to enabling workflow:***

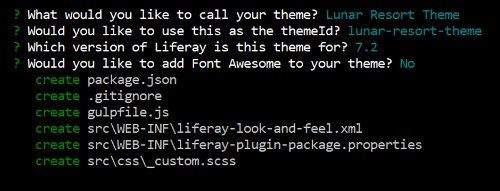
1. *Update the service layer to set each entity’s status fields.*
2. *Send the entity to Liferay DXP’s workflow framework.*
3. *Add getter methods that account for an entity’s workflow status.*
4. *Handle the entity as it returns from the workflow framework.*
5. *Update the user interface to account for workflow status.*

*Theme:*

* *Generating the theme and configuring it to extend the Atlas base theme*
* *Customizing the Header and logo*
* *Customizing the Header navigation*
* *Customizing the Footer and embedding footer navigation*
* *Creating a color scheme variant*

***Steps to generates a Theme;***

*Follow these steps to generate and configure the theme:*

1. *Install the Theme Generator. Since you’re developing a theme for Liferay DXP 7.2, install v9.x.x if it’s not installed already. Run the command below:  
   npm install -g generator-liferay-theme@9.x.x*
2. *Install the Yeoman and gulp dependencies:  
   npm install -g yo gulp*
3. *Generate the starting theme with the Theme Generator. Enter Lunar Resort Theme for the name and lunar-resort for the ID, and answer no for the Font Awesome prompt. This theme uses Clay icons instead:  
   yo liferay-theme  
     
   Figure 1: Answer no for the Font Awesome Prompt*
4. *To develop the theme you must copy the default files from the theme’s build and modify them. The /src/css/ folder and \_custom.scss file are included by default. Run the command below from the theme’s root folder to build the files:  
   gulp build*
5. *Create a new /src/templates/ folder and copy portal\_normal.ftl from the build/templates/ folder into it.*
6. *Configure the theme to extend the Atlas theme. Add a clay.scss file to the theme’s /src/css/ folder and add the import shown below:  
   @import "clay/atlas";*
7. *Create an \_imports.scss file in the /src/css/ folder and add the imports shown below to it. This includes the default imports and replaces the clay/base-variables with the Atlas base variables:  
   @import "bourbon";*

*@import "mixins";*

*@import "compat/mixins";*

1. *@import "clay/atlas-variables";*

# ***Document Repository Configuration***

*You can configure file storage in several ways. Each option is a store which can be configured through the* [*portal-ext.properties*](https://help.liferay.com/hc/en-us/articles/360028712292-Portal-Properties) *file by setting the* [*dl.store.impl= property*](https://docs.liferay.com/dxp/portal/7.2-latest/propertiesdoc/portal.properties.html#Document%20Library%20Service)*.*

*The default store is called Simple File Store. It stores* [*documents and media*](https://help.liferay.com/hc/en-us/articles/360029040531-Managing-Documents-and-Media) *files on a file system (local or mounted). The store’s default root folder is [Liferay Home]/data/document\_library.*

*The Simple File System store uses this folder path format for storing documents:*

*/companyId/folderId/numericFileEntryName/versionNumber*

*dl.store.impl=com.liferay.portal.store.file.system.AdvancedFileSystemStore*

### [**BUILT-IN MAIL SESSION PORTAL PROPERTIES**](https://help.liferay.com/hc/en-us/articles/360029031591-Configuring-Mail#built-in-mail-session-portal-properties)

If you prefer specifying your mail session offline or before deploying Liferay DXP, use portal properties.

1. Create a [portal-ext.properties file](https://help.liferay.com/hc/en-us/articles/360028712292-Portal-Properties), if you haven’t already created one.
2. Copy these default property settings into your portal-ext.properties file:  
   mail.session.mail=false

mail.session.mail.pop3.host=localhost

mail.session.mail.pop3.password=

mail.session.mail.pop3.port=110

mail.session.mail.pop3.user=

mail.session.mail.smtp.auth=false

mail.session.mail.smtp.host=localhost

mail.session.mail.smtp.password=

mail.session.mail.smtp.port=25

mail.session.mail.smtp.user=

mail.session.mail.store.protocol=pop3

mail.session.mail.transport.protocol=smtp

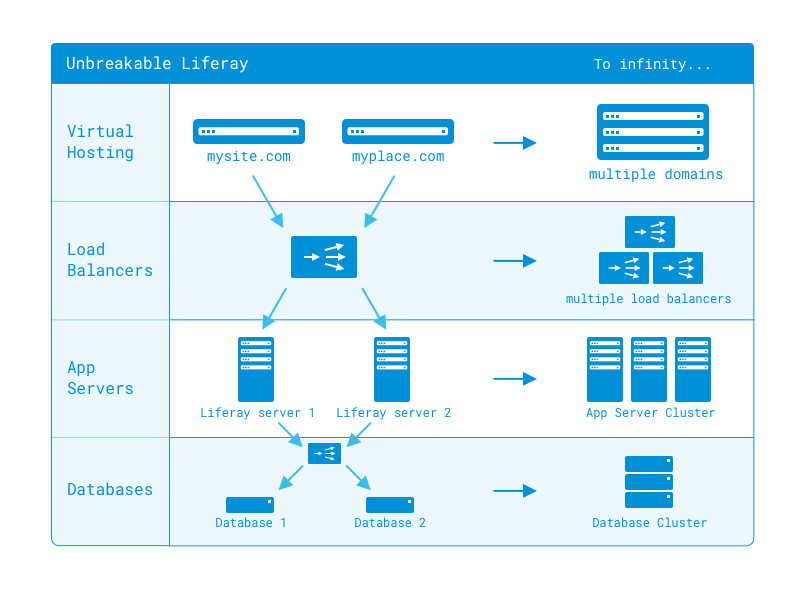
**Note:** Following emails are blacklisted by default and cannot be used in any Liferay DXP installation:

* noreply@liferay.com
* test@liferay.com
* noreply@domain.invalid
* test@domain.invalid

**Time zone configuration and default language customization are done in the Control Panel, at the Instance level.**

1. Navigate to the *Control Panel* → *Configuration*.
2. Click *Instance Settings*.
3. Click on the *Miscellaneous* tab.

**Cluster** If one server isn’t sufficient to serve your site’s high traffic needs, Liferay DXP scales to the size you need.

* Liferay DXP works well in clusters of multiple machines (horizontal cluster) or in clusters of multiple VMs on a single machine (vertical cluster), or any mixture of the two.

cluster.link.enabled=true

cluster.link.autodetect.address=www.google.com:80

### [**JDBC**](https://help.liferay.com/hc/en-us/articles/360029123851-Point-all-Nodes-to-the-Same-Liferay-DXP-Database#jdbc)

Edit your portal-ext.properties file following these steps to connect directly to your separate read and write data sources using [JDBC](https://help.liferay.com/hc/en-us/articles/360028712332-Database-Templates):

Set the default connection pool provider. For provider information, see the [JDBC properties reference](https://docs.liferay.com/portal/7.2-latest/propertiesdoc/portal.properties.html#JDBC). The default setting specifies [HikariCP](https://github.com/brettwooldridge/HikariCP) as the pool provider:  
jdbc.default.liferay.pool.provider=hikaricp

Configure JDBC connections to your separate read and write data sources. Here’s an example:  
jdbc.read.driverClassName=[place your driver name here]

jdbc.read.url=[place the URL to your "read" database here]

jdbc.read.username=[place your user name here]

jdbc.read.password=[place your password here]

jdbc.write.driverClassName=[place your driver name here]

jdbc.write.url=[place the URL to your "read-write" database here]

jdbc.write.username=[place your user name here]

jdbc.write.password=[place your password here]

1. For example JDBC connection values, please see [Database Templates](https://help.liferay.com/hc/en-us/articles/360028712332-Database-Templates).
2. Configure DXP to use the write data source (the data source whose prefix is jdbc.write.) to create the [Counter](https://docs.liferay.com/portal/7.2-latest/propertiesdoc/portal.properties.html#Counter) data source. A separate data source is always dedicated to the counter.  
   counter.jdbc.prefix=jdbc.write.

**Updating a cluster:-**

Liferay DXP supports using standard cluster maintenance techniques.

* [Rolling restarts](https://help.liferay.com/hc/en-us/articles/360028810312-Rolling-Restarts): Nodes are shut down and updated one at a time.
* [Blue-green deployment](https://help.liferay.com/hc/en-us/articles/360028810332-Blue-Green-Deployment): Blue-green involves duplicating the current environment (*blue* environment), updating the duplicate (*green* environment), and cutting over users to the updated environment (green).

# **Content Delivery Network**

A Content Delivery Network (CDN) is a network of servers deployed in multiple data centers that contain your static content. When users hit your site, that static content is loaded from a server with geographical proximity to the user, speeding up requests.

## [**Using CDN for Performance Enhancements**](https://help.liferay.com/hc/en-us/articles/360028810352-Content-Delivery-Network#using-cdn-for-performance-enhancements)

A CDN serves static web resources to users. These resources (images, CSS files, JavaScript files, etc.) are stored on multiple servers around the world. When requested, the resources are retrieved from the server nearest to the user.

**Tuning:**Tuning is the adjustment of control parameters so that a system operates in the best possible way.

Here are the tuning topics:

1)Database Connection Pool

* Deactivating Development Settings in the JSP Engine
* Thread Pool
* Many application servers’ JSP Engines are in development mode by default. Deactivate these settings prior to entering production
* Development mode is true by default.

To disable these in Tomcat, for example, update the $CATALINA\_HOME/conf/web.xml file’s JSP servlet configuration to this:

<servlet>

<servlet-name>jsp</servlet-name>

<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>

<init-param>

<param-name>development</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>mappedFile</param-name>

<param-value>false</param-value>

</init-param>

<load-on-startup>3</load-on-startup>

</servlet>

## 2)**Java Virtual Machine (JVM) tuning** primarily focuses on adjusting the garbage collector and the Java memory heap.

## [**Garbage Collector**](https://help.liferay.com/hc/en-us/articles/360028810372-Java-Virtual-Machine-Tuning#garbage-collector)

Choosing the appropriate garbage collector (GC) helps improve the responsiveness of your Liferay DXP deployment. Use the concurrent low pause collectors:

-XX:+UseParNewGC -XX:ParallelGCThreads=16 -XX:+UseConcMarkSweepGC

-XX:+CMSParallelRemarkEnabled -XX:+CMSCompactWhenClearAllSoftRefs

-XX:CMSInitiatingOccupancyFraction=85 -XX:+CMSScavengeBeforeRemark

## [**Code Cache**](https://help.liferay.com/hc/en-us/articles/360028810372-Java-Virtual-Machine-Tuning#code-cache)

Java’s just-in-time (JIT) compiler generates native code to improve performance. The default size is 48m. This may not be sufficient for larger applications. Too small a code cache reduces performance, as the JIT isn’t able to optimize high frequency methods. For Liferay DXP, start with 64m for the initial code cache size.

-XX:InitialCodeCacheSize=64m -XX:ReservedCodeCacheSize=96m

* **JAVA Heap size**

## [**Predefined Variables**](https://help.liferay.com/hc/en-us/articles/360029131871-Running-Scripts-From-the-Script-Console#predefined-variables)

**Here are the predefined variables available to scripts executed in the script console:**

* **out (java.io.PrintWriter)**
* **actionRequest (javax.portlet.ActionRequest)**
* **actionResponse (javax.portlet.ActionReponse)**
* **portletConfig (javax.portlet.PortletConfig)**
* **portletContext (javax.portlet.PortletContext)**
* **preferences (javax.portlet.PortletPreferences)**
* **userInfo (java.util.Map<String, String>)**

**There are two types of finders in Liferay:**

- Regular Finders (without permission checks)

- Filtered finders (with permission checks)

**Note: Screen names and email addresses are not interchangeable. A screen name cannot contain an @ symbol because it is used in the URL to a User’s private page.**

## [**Numeric Screen Names**](https://help.liferay.com/hc/en-us/articles/360028721712-User-Management-Additional-Topics#numeric-screen-names)

**In prior versions, numeric user screen names were disabled out of the box via the default portal property**

**users.screen.name.allow.numeric=false**

**Note:** Some permissions cannot be handled from the control panel. Asset-level permissions (for instance, permission to edit an individual blog post, or view a folder in the Documents and Media library) are managed from the individual asset

## [**Default Liferay Roles**](https://help.liferay.com/hc/en-us/articles/360028819032-Roles-and-Permissions#default-liferay-roles)

In the Roles Application appears a list of all the Roles in Liferay, by scope.

These are some of the pre-configured regular Roles:

* Guest: The Guest role is assigned to unauthenticated users and grants the lowest-level permissions.
* User: The User role is assigned to authenticated Users and grants basic permissions (mostly *Add to Page* permissions for their own Sites).
* Power User: The Power User Role grants more permissions than the User Role. It’s an extension point for distinguishing regular Users from more privileged Users. For example, you can set things up so that only Power Users have personal sites.
* Administrator: The administrator Role grants permission to manage the entire portal, including global portal settings and individual Sites, Organizations, and Users.

These are some of the pre-configured site roles:

* Site Member: The Site Member Role grants basic privileges within a Site, such as permission to visit the Site’s private pages.
* Site Administrator: The Site Administrator Role grants permission to manage *almost* all aspects of a Site including site content, site memberships, and site settings. Site Administrators cannot delete the membership of or remove roles from other Site Administrators or Site Owners. They also *cannot* assign other Users as Site Administrators or Site Owners.
* Site Owner: The Site Owner Role is the same as the Site Administrator Role except that it grants permission to manage *all* aspects of a Site, including permission to delete the membership of or remove Roles from Site Administrators or other Site Owners. They *can* assign other users as Site Administrators or Site Owners.

These are some of the pre-configured organization roles:

* Organization User: The Organization User role grants basic privileges within an Organization. If the Organization has an attached Site, the Organization User Role implicitly grants the Site member Role within the attached Site.
* Organization Administrator: The Organization Administrator Role grants permission to manage *almost* all aspects of an Organization including the Organization’s Users and the Organization’s Site (if it exists). Organization Administrators cannot delete the membership of or remove Roles from other Organization Administrators or Organization Owners. They also *cannot* assign other Users as Organization Administrators or Organization Owners.
* Organization Owner: The Organization Owner Role is the same as the Organization Administrator Role except that it grants permission to manage *all* aspects of an Organization, including permission to delete the membership of or remove Roles from Organization Administrators or other Organization Owners. They *can* assign other Users as Organization Administrators or Organization Owners.

**Tip:** It’s easy to overlook the differences between owner type roles and administrator type roles for Sites and Organizations. Site and Organization administrators *cannot* remove the administrator or owner Role from any other administrator or owner, and they *cannot* appoint other Users ad sPassword policies enforce password rules to help users

**Password policies enforce password rules to help users specify secure passwords.**

**passwords.default.policy.name=Default Password Policy**

**The structure** defines what kind of content you are creating and provides different types of fields that can be used.

**The template** defines how the elements of the structure are rendered.

**Site Templates** create a single Site structure that can be used for any new Site. They are created and administered from the Control Panel.

**Inline Content** is content that is created directly within a page. Rather than filling in fields to create content that’s added to a page later.

pages: content pages, where the content is built into the page; and widget pages, where content and other features can be added, removed, and rearranged as desired.

**A site i**s a set of pages where content or applications are published. Sites can be independent or serve as an associated organization’s website.

Liferay’s Sites provide three membership types:

**Open:** Users can become members of the Site at any time.

**Restricted:** Users can request Site membership but Site administrators must approve requests for users to become members.

**Private:** Users cannot join the Site or request Site membership. Site administrators must manually select users and assign them as Site members.

**. The difference between Sites and Organizations, of course, is that Sites organize pages, content, application data, and users (via Site memberships) whereas organizations only group users.**

If you wanted a user to have administrative access to all Sites in a Site/child Site hierarchy, you must create a role based on the Site Administrator role that has the permission *Manage SubSites*.

Any available Site templates appear for you to select. **Site templates provide** a preconfigured set of pages, applications, and content that can be used as the basis of a Site’s public or private page set.

[*Content Pages*](https://help.liferay.com/hc/en-us/articles/360028819392-Creating-Content-Pages): This new page type is flexible, especially for non-technical users. You can build pages using content created from pre-defined fragments, which themselves can contain widgets.

[*Widget Pages*](https://help.liferay.com/hc/en-us/articles/360029132251-Using-Widget-Pages): A **Widget Page is a page with a set layout that allows widgets (applications) to be displayed.**

There are two top options followed by other page types:

**Widget:** Creates a page with a layout template that defines a number of rows and columns for adding widgets to your page.

**Content:** Creates a Content Page with inline editing based on Fragments.

Below those you have other options:

**Full Page Application:** Creates a page that displays a single full page application.

**Page Set:** Creates a container for subpages that is not actually a page itself.

**Link to a Page of this Site:** Links to a page within the same Site. This is often used to make a page available in multiple parts of a Sites hierarchy.

**Panel:** A page containing any number of applications as selected by an administrator, but only one is displayed at a time. Users select the portlet they want to use from a menu on the left side of the page, and the selected portlet takes up the entire page.

**Embedded:** Displays content from another website inside your instance. An administrator can set a URL from the page management interface and that page appears in the context and within the navigation of your Liferay instance. To embed an external website, you must provide the protocol in the URL (e.g. https://www.liferay.com/).

**Link to URL:** Creates a link to any URL. This could be an external page or a link across Sites in the same Liferay instance.

**-Editable text** can be plain or rich text. Plain text has no special styling. Rich text enables text styles, typographical emphasis, alignment, and list formatting.

**Comments** are disabled by default, but administrators can enable them from *Control Panel* → *Configuration* → *System Settings* → *Pages* → *Content Page Editor*.

**A widget** is any application that you can add to a page.

Three sample layout page templates are installed by default:

* **Search:** Contains a search bar and configuration to display various facets.
* **Wiki:** Provides a page with three applications related to authoring a wiki.
* **Blog:** Provides a page with three applications related to blogging.

**The Device Simulator** (Simulation) is a powerful tool that shows you how pages look on different devices.

By default, there are four kinds of actions that can be configured for mobile families:

**Layout Template Modification:** Changes the way portlets are arranged on pages delivered to mobile devices. For example, you could have pages with more complex layouts automatically switch to a simpler template if it detects a mobile device—even if the resolution is theoretically high enough to support the standard layout.

**Theme Modification:** Selects a specific theme for different mobile device families. You’d have to have a mobile version of your Site’s theme that is automatically applied when a device hits your page.

**URL Redirect:** Sends mobile users to any URL. This can be used to direct mobile users to a mobile app download or a mobile version of the page.

**Site Redirect:** Sends mobile users to a different Site on your portal. In some cases, mobile content could be created on a mirror of your Site.

**Orphan Widgets** clears data related to widgets that have been removed from the page.

If you want to publish a piece of web content to many Sites and ensure modifications are applied to all, don’t use Site template content for that purpose. Instead, place the content in the global scope and then reference it from a *Web Content Display* application in each Site.

**By default, the following Site templates are provided:**

* **Intranet Site:** Provides a preconfigured Site for an intranet. The Home page displays the activities of the members of the Site, search, a language selector, and a list of the recent content created in the intranet. It also provides two additional pages for Documents and Media and external News obtained through public feeds.
* **Community Site:** Provides a preconfigured Site for building online communities. The Home page of a *community Site* provides message boards, search, a display of a poll and statistics of the activity of community members. The Site will also be created with a page for a wiki.

**You can disable personal Sites by adding the following properties to your portal-ext.properties file:**

layout.user.public.layouts.enabled=false

layout.user.private.layouts.enabled=false

You can allow users to create personal Sites but not have them automatically created for new users. To do this, add the following properties to your portal-ext.properties file:

layout.user.public.layouts.auto.create=false

layout.user.private.layouts.auto.create=false

If the properties layout.user.public.layouts.enabled, layout.user.private.layouts.enabled, layout.user.public.layouts.auto.create, and layout.user.private.layouts.auto.create are all set to true, which is the default, users have personal Sites and public and private pages are created automatically for new users.

You can use LARs to import data onto production servers, but you should not make this a regular occurrence. If you want to regularly move pages from one server to another, you should use Liferay DXP’s staging environment.

**Note:** LAR files are version dependent. You can’t import a LAR file that was exported from one version of Liferay into a Liferay server that’s running a different version of Liferay.

***Look and Feel Configuration* has six tabs:**

* General
* Text Styles
* Background Styles
* Border Styles
* Margin and Padding
* Advanced Styling

**Portlet widgets can communicate** with each other using public render parameters and events.

# **Configuration Templates**

Once you’ve configured a widget, Configuration Templates can save those settings in a reusable template. If someone goes in and changes the settings of a particular widget, it then becomes easy to revert those changes back to the original configuration template.

**Managing member in your site:-**

* Adding members to Sites administratively
* Adding members to Sites automatically
* Creating Teams of Site members for various functions

Since Roles are created at a global level, they cannot be created by Site Administrators

**There are two primary ways to manage and display content in Liferay DXP**

* Web Content
* Content Sets

**A Content Set** defines a list of content, and then that list can be displayed**.**

**Web content management** : It has these components:

**Web Content Editor:** A complete HTML editor for modifying fonts, adding color, inserting images, and much more.

**Structure Editor:** Define fields for structured content and more advanced designs.

**Template Editor:** Import template script files or create your own template to inform the system how to display the content within the fields determined by the structure.

**Web Content Display:** Place web content on pages in your Site.

**Asset Publisher:** Aggregate and display different types of content together in one view. This is covered in more detail in [Publishing Assets](https://help.liferay.com/hc/en-us/articles/360029133471-Publishing-Assets).

**Scheduler:** Schedule when content is reviewed, displayed or removed. This is covered in more detail in [Scheduling Web Content Publication](https://help.liferay.com/hc/en-us/articles/360029042011-Scheduling-Web-Content-Publication).

**Workflow Integration:** Run your content through a review process. This is covered in more detail in the [Workflow](https://help.liferay.com/hc/en-us/articles/360028721732-Introduction-to-Workflow) section.

**Staging:** Use a separate staging server or stage your content locally so you can keep your changes separate from the live site. This is covered in more detail in the [Staging](https://help.liferay.com/hc/en-us/articles/360028721472-Staging) section.

**Note:** Due to import/export operations, it’s possible to have both a global and a Site-scoped structure with the same structureKey. If this happens, the Site-scoped structure takes precedence, and you can’t access the global structure from that Site.

**Tags and categories** are two important tools you can use to help organize information in Liferay DXP. These tools help users to easily find the content they’re looking for through search or navigation.

Tags are an important tool that can help organize information and make it easier for users to find the content they want.

Think of tags like the index of a book and categories like its table of contents.

## [**Using Liferay’s API to Override a JSP**](https://help.liferay.com/hc/en-us/articles/360017881112-Customizing-JSPs-#using-liferays-api-to-override-a-jsp)

Here are API-based approaches to overriding JSPs in Liferay DXP:

| **Approach** | **Description** | **Cons/Limitations** |
| --- | --- | --- |
| [Dynamic includes](https://help.liferay.com/hc/en-us/articles/360017881132-Customizing-JSPs-with-Dynamic-Includes-) | Adds content at dynamic include tags. | Limited to JSPs that have dynamic-include tags (or tags whose classes inherit from IncludeTag). Only inserts content in the JSPs at the dynamic include tags. |
| [Portlet filters](https://help.liferay.com/hc/en-us/articles/360018159931-JSP-Overrides-Using-Portlet-Filters-) | Modifies portlet requests and/or responses to simulate a JSP customization. | Although this approach doesn’t directly customize a JSP, it achieves the effect of a JSP customization. |

## [**Overriding a JSP Without Using Liferay’s API**](https://help.liferay.com/hc/en-us/articles/360017881112-Customizing-JSPs-#overriding-a-jsp-without-using-liferays-api)

| **Approach** | **Description** | **Cons/Limitations** |
| --- | --- | --- |
| [OSGi fragment (deprecated as of Liferay 7.0)](https://help.liferay.com/hc/en-us/articles/360017881152-JSP-Overrides-Using-OSGi-Fragments-) | Completely overrides a module’s JSP using an OSGi fragment | Changes to the original JSP or module can cause runtime errors. |
| [Custom JSP bag (deprecated as of Liferay 7.0)](https://help.liferay.com/hc/en-us/articles/360017881172-JSP-Overrides-Using-Custom-JSP-Bag-) | Completely override a Liferay DXP core JSP or one of its corresponding -ext.jsp files. | For Liferay DXP core JSPs only. Changes to the original JSP or module can cause runtime errors. |

bnd.bnd : bnd file is MANIFEST configuration template file which used to auto create MANIFEST.MF, so developers are not required to focus on MANIFEST file creation

* It also used to activate Bundle with BundleActivator configuration if we are not going to use OSGI Declarative Services.

# **Overriding lpkg files**

1. Shut down Liferay DXP.
2. Create a folder called override in the Liferay DXP instance’s osgi/marketplace folder.
3. Name your updated .jar the same as the .jar in the original .lpkg, minus the version information. For example, if you’re overriding the com.liferay.amazon.rankings.web-1.0.5.jar from the Liferay CE Amazon Rankings.lpkg, you’d name your .jar com.liferay.amazon.rankings.web.jar.
4. Copy this .jar into the override folder you created in step one.

This works for applications from Marketplace, but there’s also the static .lpkg that contains core Liferay technology and third-party utilities (such as the servlet API, Apache utilities, etc.). If you find you need to customize or patch any of these .jar files, deploying these customizations is a similar process:

1. Make your customization and package your .jar file.
2. Name your updated .jar the same as the original .jar, minus the version information. For example, a customized com.liferay.portal.profile-1.0.4.jar should be com.liferay.portal.profile.jar.
3. Place this .jar in the osgi/static folder.

# **Introduction to Asset Framework**

Liferay’s asset framework is a system that allows you to add core Liferay features to your application. For example, if you’ve built an event management application that displays a list of upcoming events, you can use the asset framework to let users add tags, categories, or comments to make entries more self-descriptive.

# **Building a Site**

A site is a set of pages where content or applications are published. Sites can be independent or serve as an associated organization’s website. You can create as many different sites as you like within the context of a single Liferay instance.

Liferay’s Sites provide three membership types:

**Open:** Users can become members of the Site at any time.

**Restricted:** Users can request Site membership but Site administrators must approve requests for users to become members.

**Private:** Users cannot join the Site or request Site membership. Site administrators must manually select users and assign them as Site members.

[*Content Pages*](https://help.liferay.com/hc/en-us/articles/360028819392-Creating-Content-Pages): This new page type is flexible, especially for non-technical users. You can build pages using content created from pre-defined fragments, which themselves can contain widgets.

[*Widget Pages*](https://help.liferay.com/hc/en-us/articles/360029132251-Using-Widget-Pages): Liferay DXP’s traditional page type is made up of one or more widgets. There are some use cases (particularly if a page’s sole purpose is to host an application) to prefer widget pages.

**Layouts** are special Sections that define spaces where you can add fragments or widgets. Each layout you add fills the width of the page. You can add any number of layouts to the page.

Publishing RSS Feeds

RSS is a family of web feed formats used to publish frequently updated works such as blog entries and news articles. **RSS allows users to stay up-to-date with your site’s content without actually having to visit your site**. Users can use their own RSS feed readers to aggregate content, and you can also use RSS to share and aggregate content across sites. Next, you’ll see how to create RSS feeds from Asset Publisher configurations.

## [**Which Form Builder Should I Use?**](https://help.liferay.com/hc/en-us/articles/360029134031-Introduction-to-Forms#which-form-builder-should-i-use)

Liferay Forms (also referred to as Forms) is a relatively new application, first appearing in Liferay DXP version 7.0. If you can use Liferay Forms for your use case, you should.

So the question “Which form builder should I use?” can be restated to “When should I use Dynamic Data Lists?”

* Use Dynamic Data Lists (DDL) if you need a way for users to enter data, *and* you need to display the data in the user interface.
* **Use DDL if you need to style your lists and forms with templates.**
* Use DDL if there’s a field type you need that’s not included (yet) in Liferay Forms. These are the field types included in DDL that *are not* in Liferay Forms at the time of this writing: - **Color - Geolocation - Web Content - Link to Page**