# Cahier de TP
# « Liferay 7.x - Développer un portail d'entreprise»

Pré-requis :
Poste développeur avec accès réseau Internet libre
Linux (Recommandé) ou Windows 10
Pré-installation de :

- Git
- JDK11+
- npx, npm
- IDEs : IntelliJ

## Table des matières

## Présentation projet fil rouge :

Progressivement, nous allons construire un mini-portail **“EventPortal”** composé :

- d’un site dédié
- de pages de navigation
- de contenus CMS structurés
- d’un mécanisme de staging opérationnel
- d’une portlet custom
- d’une UI modernisée Clay
- d’une entité Service Builder (“Event”)
- de permissions fines
- d'extensions OSGi pour personnalisation

# Atelier 1: Installation Liferay 7.4 + Base de données

Objectifs

- installer un bundle Liferay 7.4 avec une base de production

## *1.1 Installation d'une base Postgres*

Démarrer un serveur postgres et sa console d'administration

```
docker compose -f postgres-docker-compose.yml up -d
```

Accéder à localhost:81 et se logger avec admin@admin.com/admin

Enregistre un serveur avec comme paramètres de connexion :

- host : ***liferay-postgresql***
- user : ***postgres***
- password : ***postgres***

Créer :

- une base ***liferay***
- un utilisateur ***liferay*** avec tous les privilèges

```
CREATE USER liferay_user WITH PASSWORD 'liferay';

CREATE DATABASE liferay
  WITH OWNER liferay_user
  ENCODING 'UTF8'
  CONNECTION LIMIT -1;


GRANT ALL PRIVILEGES ON DATABASE liferay TO liferay_user;
```

## *1.2 Démarrage Bundle Tomcat*

Télécharger la distribution Liferay 7.4 (bundle Tomcat) et l'extraire dans un répertoire de travail

Placer le fichier ***portal-ext.properties*** dans le répertoire d'extraction avec :

```
jdbc.default.driverClassName=org.postgresql.Driver
jdbc.default.url=jdbc:postgresql://localhost/liferay
jdbc.default.username=liferay_user
jdbc.default.password=liferay
```

Démarrer Tomcat

```
./tomcat/bin/startup.sh
```

Surveiller le démarrage :

```
tail -f ./tomcat/logs/catalina.out
```

**Vérifications** :

- ***Accès :*** `http://localhost:8080`, Vous devez voir la page d'accueil de Liferay 7.4 (assistant de configuration).

- Base de données : Visualiser les tables créées

## *1.2 Assistant de configuration initiale*

1. Dans le navigateur, suivez les écrans de l'assistant :
    - choisissez la **langue** (par ex. Français),
    - définissez le **nom du portail** comme ***EventPortal***
    - créez le **compte administrateur** (adresse e-mail, mot de passe).
2. Effectuer un redémarrage et connectez-vous avec le compte administrateur créé.

Vérifications **:**
Accès à la page d'accueil du portail, avec le menu d'administration (icône "roue dentée" ou "Produits") disponible pour votre utilisateur.

# Atelier 2 : Sites, Pages et Navigation

Objectifs :

- Découvrir la gestion des pages dans Liferay.
- Comprendre la différence entre *Page de contenu* et *Page widget*.
- Construire la structure de navigation d'un site d'entreprise.
- Manipuler les URLs conviviales (*Friendly URLs*) utilisées pour le référencement et l'ergonomie.
- Préparer l'architecture du site qui servira de support à tout le fil rouge.

## *2.1 Lister les sites*

Commencer par lister les site

***Menu > Panneau de contrôle > Sites***
2 sites doivent apparaître :

- **EventPortal**
- **global :** Ce site est un site technique utilisé pour rassembles les assets réutilisables

## *2.2 Création de pages*

Nous voulons créer différents types de pages :

- ***Home*** : La page d'accueil
- ***Recherche*** : La page Widget permettant d'afficher les résultats d'une recherche full-text
- ***Actualités*** : Une page Widget permettant le rendu de portlets, elle contient une sous-page :
  - ***Archives***
- ***Évènements*** : Un ensemble de page contenant les pages :
  - ***Présentation***
  - ***Liste des évènements***
  - ***Ajouter un événement***
- ***Contact*** : Une page Widget

Aller dans ***"Créateur de site" → "Pages"***

Configurer la page Home existante

- Nom : **Home**
- Friendly URL : /home

Pour la page Actualités

1. **Ajouter → Page de widget**
2. Nom : **Actualités**

Dans général :

- Disposition : 1 colonne
- URL conviviale : **`/actualites`**

La page Événements (Ensemble de pages)

1. **Ajouter → Page vierge**
2. Nom : **Événements**
3. **Ensemble de pages**
4. Cliquer sur **Publier** sans ajouter de contenu pour le moment

URL conviviale : **`/evenements`**

Créer ensuite le sous-pages :

- Présentation / Contenu
- Liste / Widget
- Ajouter  / Widget

La page Contact (Widget)

1. Ajouter → **Page de widget**
2. Nom : **Contact**

Disposition : 1 colonne
URL conviviale : **`/contact`**

## *2.3 Organiser la navigation*

Depuis **Créateur de site → Pages** :

*Modifier l'ordre*

Glisser-déposer les pages pour obtenir :

1. Home
2. Actualités
3. Événements
4. Contact
5. Recherche (si créée maintenant)

*Créer une sous-page*

1. Sélectionner **Événements**
2. Ajouter → **Page de widget**
3. Nom : **Archives**
4. Publier

*Définir les pages publiques / privées*

Via **Configurer :**

- Contact ou Événements → Supprimer la permission Voir au rôle Guest

# Atelier 3 : Fonctions CMS : contenus, structures & templates

Objectifs

- Mettre à jour une page de contenu du portail (Home ou Présentation des Événements)
- Créer un contenu structuré avec une *Structure* et un *Template*
- Publier une actualité conforme à ce modèle
- Afficher un contenu :
    - via un widget Web Content Display,
    - via un Fragment Web Content,
    - ou via une Display Page (méthode moderne recommandée)
- Utiliser les Documents & Media pour intégrer une image dans une page

## *3.1 Mise à jour d'un contenu statique*

Essayer l'éditeur de CMS pur sur la page d'accueil par exemple :

- Ajout de texte
- Ajout d'image

## *3.2. Créer une structure nommé Actualité*

**Contenu → Contenu web → Structures**

- Nom : **Actualité**

Champs :

- *titre (text)*
- *imagePrincipale (image)*
- *accroche (text)*
- *texte (rich text)*
- *datePublication (date)*

Pour chaque champ, utiliser l'onglet avancé pour fixer l'identifiant du champ.

## *3.2 Créer un gabarit Freemarker*

Activer l'onglet Modèle de document et créer un modèle associé par défaut à la structure Actualité

Nom du template : **Actualité par défaut FTL**

```
<h1>${titre.getData()}</h1>
<img src="${imagePrincipale.getData()}" />
<p>${accroche.getData()}</p>
```

```
<div>${texte.getData()}</div>
<p>Publié le ${datePublication.getData()}</p>
```

## *3.3 Créer un Web Content utilisant la structure*

- Ajouter un article “Actualité numéro 1”
- Remplir les champs
- Publier

## *3.4. Afficher l’article dans une page*

**Méthode 1 : via Web Content Display (classique)**

1. Naviguer dans la **Pages → Actualités**
2. Activer le bouton + pour ajouter du contenu
3. Sélectionner **Affichage de contenu web**
4. Configurer en sélectionnant l’actualité précédente
5. Enregistrer / Publier

**Méthode 2 : via Fragment “Web Content”**

1. Modifier la page Home
2. Ajouter un **fragment → Web Content**
3. Sélectionner le contenu créé
4. Publier

**Méthode 3 : via Modèle de page d’affichage**

Créer un modèle de page d’affichage

1. Aller dans ***Conception → Modèles de page → Modèle de page d’affichage***
2. Ajouter un modèle, le nommer : **Actualité – Page d’affichage**
3. Utiliser des fragments dynamiques pour afficher les champs de la structure Actualités
4. Ajouter un bouton **Retour aux actualités**
5. Publier

Associer le Modèle de page d’affichage à la structure

1. Retourner dans votre structure actualité
2. Cliquer sur ***Détails → Affichage → Page de visualisation***
3. Sélectionner : *Actualité – Page d’affichage*
4. Enregistrer

Définir une liste de contenus **Actualités archivées**

1. ***Créateur de sites → Liste de contenus***

2. ***Nouveau → Collection Dynamique*** **: Actualités archivées**
3. ***Contenu web → Actualités***
4. **Éventuellement les autres champs**

Sur la page *Archives* :

1. Ajouter un widget **Agrégateur de contenu (Asset Publisher)**
2. Choisir la collection que l'on vient de créer
3. Créer une nouvelle actualité et vérifier que le modèle de page d'affichage y est bien associé
4. Accéder à la page archive et Cliquer sur un élément

# Atelier 4 : Tables fondamentales de Liferay

Objectifs

À la fin de ce TP, les stagiaires sauront :

- Identifier les tables essentielles de Liferay
- Comprendre la relation entre sites, pages, utilisateurs, rôles et contenus
- Exécuter des requêtes SQL pour inspecter la structure interne d'une installation Liferay
- Retrouver les identifiants internes indispensables pour diagnostiquer un problème ou croiser des données
- Comprendre la différence entre **Content Page**, **Widget Page**, **Layout**, **LayoutSet**, **JournalArticle**, **DLFileEntry**, etc.

## *4.1. Les sites*

Les sites dans Liferay sont stockés dans la table :

```
SELECT
  g.groupId,
  g.companyId,
  g.friendlyURL,
  c.name AS company_name
FROM "group_" g
JOIN "company" c ON c.companyId = g.companyId
WHERE g.site = TRUE
ORDER BY g.groupId;
```

Utilisé le ***groupId*** dans la suite des requêtes

## *4.2 Utilisateurs et rôles*

Utilisateurs

```
SELECT userId, screenName, emailAddress, firstName, lastName,
status
FROM "user_"
ORDER BY userId;
```

Rôles

```
SELECT roleId, name, type_
FROM "Role_"
ORDER BY roleId;
```

Interprétation `Role_.type_`

| type_ | Signification |
|---|---|
| 1 | Site Role |

| type_ | Signification |
|---|---|
| 2 | Organization Role |
| 3 | Regular Role |
| 4 | Depot Role |

Association User/Rôle

```
SELECT u.screenName, r.name AS roleName
FROM "users_roles" ur
JOIN "User_" u ON u.userId = ur.userId
JOIN "Role_" r ON r.roleId = ur.roleId
ORDER BY u.screenName;
```

### *4.3 Les pages*

Les pages sont stockées dans :

- **`Layout`** → une ligne par *page*
- **`LayoutSet`** → Conteneurs de pages 2 par sites
- Permissions

Pages

```
SELECT layoutId, plid, parentLayoutId, name, friendlyURL, type_,
hidden, priority
FROM "layout"
WHERE groupId = groupId
  AND privateLayout = FALSE
ORDER BY parentLayoutId, priority;
```

Interprétation :

- `type_` :
  - `content` → Page de contenu
  - `portlet` → Page de widgets
  - `asset_display` → Modèle de page de visualisation
  - `utility` → Page système
  - `node` → Ensemble de page
- `friendlyURL` → /home, /actualites…
- `parentLayoutId` → hiérarchie
- `priority` → ordre

Les conteneurs de page

```
SELECT layoutId, plid, parentLayoutId, name, friendlyURL, type_,
hidden_, priority
```

```
FROM "layout"
WHERE groupId = <groupId>
  AND privateLayout = FALSE
ORDER BY parentLayoutId, priority;
```

Tu verras deux lignes :

- `privateLayout = false` → pages publiques
- `privateLayout = true` → pages privées (même si tu n'en utilises plus en 7.4)

Association Layout / LayoutSet

```
SELECT
  l.layoutId,
  l.friendlyURL,
  l.type_,
  ls.layoutSetId,
  ls.privateLayout,
  ls.themeId
FROM "layout" l
JOIN "layoutset" ls
  ON ls.groupId = l.groupId
 AND ls.privateLayout = l.privateLayout
WHERE l.groupId = <groupId>;
```

Permissions sur les pages

Les permissions sont stockées dans :

- **`ResourcePermission`** → qui peut faire quoi sur quoi
- **`ResourceAction`** → quelles actions existent
- **`Role_`** → les rôles
- **`Layout`** → les pages

Exemple : page /actualites

```
SELECT
    rp.roleId,
    r.name AS roleName,
    rp.actionIds,
    rp.primKey
FROM "resourcepermission" rp
JOIN "Role_" r ON r.roleId = rp.roleId
WHERE rp.name = 'com.liferay.portal.kernel.model.Layout'
  AND rp.primKey LIKE (
```

```
        SELECT CAST(plid AS VARCHAR)
        FROM "Layout"
        WHERE friendlyURL='/actualites'
          AND groupId = <groupId>
  );
```

*actionIds* est un *bit mask* de la table resourceaction

Plus lisible :

```
SELECT
  l.plid,
  l.friendlyURL,
  l.name AS page_name,
  r.roleId,
  r.name AS roleName,
  string_agg(ra.actionId, ', ' ORDER BY ra.actionId) AS
allowedActions
FROM "resourcepermission" rp
JOIN "role_" r
  ON r.roleId = rp.roleId
JOIN "layout" l
  ON l.plid::text = rp.primKey
JOIN "resourceaction" ra
  ON ra.name = rp.name
 AND (rp.actionIds & ra.bitwiseValue) <> 0
WHERE rp.name = 'com.liferay.portal.kernel.model.Layout'
  AND rp.scope = 4              -- permissions individuelles
  AND l.groupId = 20117         -- site Event Portal
  AND l.privateLayout = FALSE   -- pages publiques
GROUP BY
  l.plid,
  l.friendlyURL,
  l.name,
  r.roleId,
  r.name
ORDER BY
  l.friendlyURL,
  r.name;
```

### *4.4. Contenus : Web Contents + leurs structures*

Lister les contenus

```
SELECT articleId, urltitle, version, status, userName
FROM "journalarticle"
WHERE groupId = 20117;
```

Association structure contenu

```
SELECT ja.articleId, ja.urltitle, ds.name
FROM "journalarticle" ja
JOIN "ddmstructure" ds ON ds.structureid = ja.ddmstructureid
WHERE ja.groupId = 20117;
```

### *4.5 Documents & Media*

Documents stockés dans :

- **DLFileEntry :** Un document
- **DLFileVersion :** Versionning
- **DLFolder :** Un répertoire

```
SELECT fileEntryId, title, mimeType, groupId, folderId
FROM "dlfileentry"
WHERE groupId = 20117;
```

# Atelier 5 : Gestion de la configuration

Objectif

Tester les 3 techniques de configuration :

- **portal-ext.properties**
- **OSGi .config**
- **modification via System Settings**

## *5.1 Propriétés de portal-ext*

Ajouter dans ***portal-ext.properties*** :

```
theme.css.fast.load=false
```

Redémarrer et constater le changement.

Observation attendue

- Les CSS ne sont plus concaténées/minifiées
- Le chargement front est plus verbeux (utile en debug)

## *5.2. Configuration OSGi*

Configuration via l'UI

***Panneau de contrôle > Paramètres Système > Contenu web***

- Activer/désactiver "Enable comments"

Observation

- Le changement est immédiat
- Aucun redémarrage requis

Analyse

- La modification met à jour la table configuration

```
SELECT
  configurationId,
  dictionary
FROM configuration_;
```

Configuration via fichiers :

Vérifier la valeur de la case à cocher « Index de toutes les versions des articles activé »

Sortir de la page de configuration

Créer le fichier :

`com.liferay.journal.configuration.JournalServiceConfiguration.config`

Avec comme contenu :

```
indexAllArticleVersionsEnabled="false"
```

Retrouver l’effet dans l’UI

## *5.3 Verbosité des logs*

***Panneau de contrôle | Système | Administration du serveur***

Visualiser les loggers

Passer ***com.liferay*** en mode DEBUG et visualiser l’effet sur ***catalina.out***

# Atelier 6 : Exploitation des modules OSGI

Objectifs

- Utiliser la Gogoshell
- Lister les modules OSGi
- Identifier un module Liferay
- Arrêter et redémarrer un module
- Observer l'impact immédiat dans l'interface utilisateur

Accéder au Gogo Shell

*Panneau de contrôle → Système → Gogo Shell*

Taper :

```
help
lb -s
```

Identifier l'ID du bundle ***com.liferay.journal.web, de***

Arrêter le module :

```
stop <ID_DU_BUNDLE>
```

Observer l'impact dans le menu Contenu de l'administration de site

```
start <ID_DU_BUNDLE>
```

Lister les services de type

```
services -b <ID_DU_BUNDLE>
```

Lister les services de type *ModelResourcePermission*

```
services
com.liferay.portal.kernel.security.permission.resource.ModelResour
cePermission
```

Lister tous les composants :

```
scr:list
```

Voir les dépendances d'un composant :

```
scr:info
com.liferay.portal.security.permission.internal.resource.PortletRe
sourcePermissionImpl
```

# Atelier 7 : Staging + Export/Import

Objectif

Mettre en place du staging, publier un contenu, tester import/export.

## *7.1 Staging*

Activer le staging

Menu du site  > En cours de publication > Organiser

- Choisir **Local Staging**
- Sélectionner pages & contenus à mettre en staging

Ajouter du contenu dans le staging

- Editer la page Home et ajouter une “Bannière Cookie”
- Publier sur le staging
- Comparer le staging (pré-production) et le live (en utilisant un autre navigateur ou un onglet privé)

Publier sur le vrai site

- Publier sur le site
- Vérifier l’apparition dans le site live

## *7.2 Export / Import*

- Exporter le site dans un `.lar`
- Importer dans un nouveau site “EventPortal_Copy”

Aller sur le vrai site, effectuer un export puis télécharger le fichier .lar

Créer une nouveau site « Copie »

Effectuer ensuite une importation sur le site « Copie »

Retrouver les pages

# Atelier 8 : Développement Portlet MVC simple

Objectif

- Créer une portlet “Hello”.

## *8.1 Mise en place du workspace*

Installer blade cli

https://learn.liferay.com/w/dxp/development/tooling/blade-cli

Vérifier avec

```
blade -v
```

Vous placer dans un répertoire de travail et exécuter :

```
blade init liferay-workspace
```

## *8.2 Import du workspace dans l’IDE*

1. Ouvrez votre IDE (Eclipse/Liferay Developer Studio ou IntelliJ).
2. Importez le workspace :
    - Eclipse/Liferay Developer Studio :
        - **File > Import… > Existing Gradle Project**,
        - sélectionnez le dossier `my-liferay-workspace`.
    - IntelliJ :
        - **Open…**,
        - sélectionnez `my-liferay-workspace` et laissez IntelliJ détecter le projet Gradle.
3. Attendez la fin de l’indexation et de la synchronisation Gradle.

Contrôle :
Dans la vue de projets, vous voyez le projet `my-liferay-workspace` et son sous-répertoire `modules` (vide pour l’instant).

## *8.3 Création d’un module portlet*

Créer le module MVC

```
cd liferay-workspace
blade create -t mvc-portlet -p org.formation.hello -c HelloPortlet
hello
```

- `-t mvc-portlet` : type de module,
- `-p org.formation.hello` : package Java,

- `-c HelloPortlet` : nom de la classe portlet,
- `hello` : nom du module.

Vérifier la création du module dans votre IDE

Vérifiez dans l'IDE :

- un nouveau projet `hello` apparaît sous le dossier `modules`,
- la classe `HelloPortlet` existe dans `src/main/java/org/formation/hello`.

Ouvrez la vue JSP principale :
`hello/src/main/resources/META-INF/resources/view.jsp`.

Remplacez le contenu de `view.jsp` par quelque chose de simple, par exemple :

```
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil" %>
    <%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
    <%@ taglib uri="http://liferay.com/tld/theme" prefix="liferay-theme" %>

    <liferay-theme:defineObjects />
    <portlet:defineObjects />

    <h1>Hello Liferay 7.4 !</h1>
    <p>
        Ce portlet a été créé dans un module OSGi à l'aide de Blade CLI.<br/>
        Utilisateur connecté : <strong><%=
    HtmlUtil.escape(themeDisplay.getUser().getFullName()) %></strong>
    </p>
```

Contrôle :

Le code compile dans l'IDE (pas d'erreur rouge dans la classe ni dans la JSP).

Déployer

Configurer la cible de déploiement en ajoutant dans le fichier gradle.properties du workspace :

```
liferay.workspace.home.dir=/chemin/vers/ton/liferay
```

Assurer vous que le serveur est démarré

Puis effectuer la commande de déploiement dans le répertoire du workspace

```
./gradlew :modules:hello-web:deploy
OU
./gradlew deploy
```

Contrôle :

Dans le répertoire `deploy` du Liferay Home, le JAR du module `hello-event` doit apparaître, puis être déplacé automatiquement une fois déployé.

Dans les traces du serveur, on doit pouvoir voir la ligne :

```
Started org.formation.helloEvent_1.0.0
```

## *8.4. Ajouter la portlet*

1. Connectez-vous au portail avec votre compte administrateur
2. Dans la page Home de notre portail .
3. Dans le menu d'ajout d'applications, recherchez le portlet :
    - nom de la catégorie : selon la configuration du module (souvent "Exemple" ou "hello"),
    - nom du portlet : **HelloPortlet** ou libellé défini dans le bundle de ressources.
4. Ajoutez le portlet à la page.
5. Vérifiez l'affichage :
    - le message "Hello Liferay 7.4 !" apparaît,
    - le nom de l'utilisateur connecté est affiché.

Contrôle final :

Si le portlet s'affiche correctement et montre votre nom d'utilisateur connecté, le TP est réussi.

# Ateliers 9 : Développement d’une Portlet MVC avec validation

Objectifs

- Implémenter un `MVCActionCommand`
- Effectuer la **validation serveur**
- Gérer des erreurs utilisateur (`SessionErrors`)
- Naviguer entre écrans via `mvcPath`

## *9.1 Création du module*

Depuis le workspace :

```
blade create -t mvc-portlet \
  -p org.formation.event \
  -c EventPortlet \
  event
```

Modifier `view.jsp`

Créer une zone d'affichage + lien vers l’écran d’édition :

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

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


<h1>Liste des événements</h1>

<portlet:renderURL var="editURL">
   <portlet:param name="mvcPath" value="/edit.jsp"/>
</portlet:renderURL>

<a href="${editURL}">Ajouter un événement</a>
```

Créer `edit.jsp`

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>

<h2>Ajouter un événement</h2>
```

```
<liferay-ui:error key="title-required" message="Le titre est
obligatoire"/>

<portlet:actionURL name="saveEvent" var="saveURL" />

<form action="${saveURL}" method="post">
   <label>Titre :</label>
   <input type="text" name="title" />
   <button type="submit">Enregistrer</button>
</form>
```

## *9.2 Implémenter l'Action Command*

Créer :

`src/main/java/org/formation/event/action/SaveEventMVCActionCommand.java`

```
@Component(
    property = {
        "javax.portlet.name=org_formation_event_EventPortlet",
        "mvc.command.name=saveEvent"
    },
    service = MVCActionCommand.class
)
public class SaveEventMVCActionCommand implements MVCActionCommand {

    @Override
    public boolean processAction(ActionRequest req, ActionResponse resp) {

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

        if (Validator.isNull(title)) {
            SessionErrors.add(req, "title-required");
            resp.setRenderParameter("mvcPath", "/edit.jsp");
            return false;
        }

        SessionMessages.add(req, "event-saved");
        return true;
    }
}
```

Tester dans Liferay

1. Déployer :

```
./gradlew :modules:event:deploy
```

2. Ajouter la portlet sur une page

3. Cliquer *Ajouter une entrée*
4. Tester :
    - Formulaire vide → message d’erreur
    - Formulaire rempli → succès

# Atelier 10 : AUI et Navigation

Objectifs :

- Utiliser AlloyUI
- Navigation Simple

## *10.1 AlloyUI*

Implémenter un formulaire AlloyUI

Dans `edit.jsp, remplacer le formulaire par` :

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

<portlet:actionURL name="saveEvent" var="saveURL" />

<aui:form action="${saveURL}" method="post" name="fm">
    <aui:input name="title" label="Titre" />

    <aui:button-row>
        <aui:button type="submit" value="Enregistrer" />
    </aui:button-row>
</aui:form>
```

Liste avec SearchContainer

Pour l'instant, les événements sont **mockés** (liste Java en mémoire).

Dans `view.jsp` :

```
<%@ page import="java.util.*" %>
<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>



<liferay-ui:search-container
    total="<%= events.size() %>"
    emptyResultsMessage="Aucun événement">

    <liferay-ui:search-container-results
        results="<%= events.subList(searchContainer.getStart(),
searchContainer.getEnd() > events.size() ? events.size() :
searchContainer.getEnd()) %>" />
```

```
    <liferay-ui:search-container-row
        className="java.lang.String"
        modelVar="event">

        <liferay-ui:search-container-column-text
            name="Nom"
            value="<%= event %>" />
    </liferay-ui:search-container-row>

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

## *10.2 Navigation simple*

Créer une page home.jsp comme suit :

```
<%@ page import="java.util.*" %>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8" %>

<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet"
%>

<h1>Bienvenue sur la page d'accueil des événements</h1>
<portlet:renderURL var="listURL">
     <portlet:param name="mvcRenderCommandName"
value="/event/list" />
</portlet:renderURL>
<a href="<%= listURL %>">Afficher la liste des événements</a>
```

Faire en sorte que ce soit la page d'accueil du portlet en modifiant les paramètres de la classe portlet :

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

La page propose un lien utilisant ***mvcRenderCommandName***

Implémenter un MVCRenderCommand

Dans une classe EventListMVCRenderCommand

```
@Component(
        property = {

"javax.portlet.name=org_formation_event_EventPortlet",
                "mvc.command.name=/event/list"
        },
        service = MVCRenderCommand.class
```

```
)
    public class EventListMVCRenderCommand implements
MVCRenderCommand {
    @Override
    public String render(RenderRequest renderRequest,
RenderResponse renderResponse) throws PortletException {
        // Liste Mockée
        List<String> events = Arrays.asList(
                "Conférence Liferay",
                "Atelier Dev",
                "Meetup Architecture"
        );

        renderRequest.setAttribute("events", events);


        return "/view.jsp";
    }
}
```

Tester dans Liferay

1. Déployer :

```
./gradlew :modules:event:deploy
```

2. Vérifier le déploiement dans les traces
3. Visualiser l’affichage de la liste

# Atelier 11 : Couche service via ServiceBuilder

Objectifs :

- Créer un modèle `Event` via Service Builder
- Implémenter `addEvent` et `getEvents`
- Exposer la liste via REST Builder
- Tester avec Postman / curl
- Appeler le service depuis une portlet ou une Remote App

## *11.1 Création du module ServiceBuilder*

```
blade create -t service-builder -p org.formation.event events
```

2 modules générés :

- `events-api`
- `events-service` contient `service.xml`

Mettre à jour le fichier `service.xml` :

```
<service-builder dependency-injector="ds" package-path="org.formation.event">
 <namespace>EVENT</namespace>
 <!--<entity data-source="sampleDataSource" local-service="true" name="Foo"
remote-service="false" session-factory="sampleSessionFactory" table="foo" tx-
manager="sampleTransactionManager uuid="true"">-->
 <entity name="Event" local-service="true" remote-service="false">

 <!-- PK fields -->

 <column name="eventId" 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" />

 <!-- Other fields -->

 <column name="title" type="String" />
 <column name="dateDebut" type="Date" />
 <column name="dateFin" type="Date" />

 <!-- Order -->

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

```
            </order>

            <!-- Finder methods -->

            <finder name="DateDebut" return-type="Collection">
                  <finder-column name="groupId" />
                  <finder-column name="dateDebut" />
            </finder>

            <!-- References -->

            <reference entity="AssetEntry" package-
path="com.liferay.portlet.asset" />
            <reference entity="AssetTag" package-
path="com.liferay.portlet.asset" />
      </entity>
</service-builder>
```

Générer les classes

```
./gradlew buildService
```

Visualiser les classes générées

## *11.2 Implémentation de LocalServiceImpl*

EventLocalServiceImpl.java :

```
public Event addEvent(long userId, long groupId, String title,
Date dateDebut, Date dateFin, ServiceContext sc) {

 long id = counterLocalService.increment(Event.class.getName());
 Event event = eventPersistence.create(id);

 event.setGroupId(groupId);
 event.setCompanyId(sc.getCompanyId());
 event.setUserId(userId);
 event.setTitle(title);
 event.setDateDebut(dateDebut);
 event.setDateFin(dateFin);

 return eventPersistence.update(event);
}


public List<Event> getEvents(long groupId, int start, int end) {
     return eventPersistence.findByGroupId(groupId,  start, end);
}

public int getEventsCount(long groupId) {
     return eventPersistence.countByGroupId(groupId);
}
```

Déployer le service

```
./gradlew deploy
```

Vérifier que le bundle est démarré

## *11.3 Intégration Portlet*

Dans le projet du portlet, ajouter une dépendance

```
compileOnly project(":modules:events:events-api")
```

Modifier MVRenderCommand

```
@Reference
private org.formation.event.service.EventLocalService
_eventLocalService;

@Override
public String render(RenderRequest renderRequest, RenderResponse
renderResponse) throws PortletException {
    ThemeDisplay themeDisplay =
            (ThemeDisplay)
renderRequest.getAttribute(WebKeys.THEME_DISPLAY);

    List<org.formation.event.model.Event> events =
            _eventLocalService.getEvents(
                    themeDisplay.getScopeGroupId(), -1, -1);

    System.out.println("Rendering Event List with " +
events.size() + " events.");
    renderRequest.setAttribute("events", events);


    return "/view.jsp";
}
```

Modifier MVCActionCommand

```
@Reference
private org.formation.event.service.EventLocalService
_eventLocalService;

@Override
public boolean processAction(ActionRequest req, ActionResponse
resp) {

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

```
    System.out.println("Event Title: " + title);
    if (Validator.isNull(title)) {
        SessionErrors.add(req, "title-required");
        resp.setRenderParameter("mvcPath", "/edit.jsp");
        return false;
    }

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

    ServiceContext serviceContext =
            null;
    try {
        serviceContext = ServiceContextFactory.getInstance(
                org.formation.event.model.Event.class.getName(),
                req);
    } catch (PortalException e) {
        throw new RuntimeException(e);
    }


    DateFormat dateFormat =
            DateFormatFactoryUtil.getSimpleDateFormat("yyyy-MM-
dd");

    Date dateDebut =
            ParamUtil.getDate(req, "dateDebut", dateFormat);

    Date dateFin =
            ParamUtil.getDate(req, "dateFin", dateFormat);

    _eventLocalService.addEvent(
            themeDisplay.getUserId(),
            themeDisplay.getScopeGroupId(),
            title,
            dateDebut,
            dateFin,
            serviceContext
    );

    SessionMessages.add(req, "event-saved");
    return true;
}
```

Déployer et vérifier le déploiement

Accéder au servlet, enregistrer un événement et vérifier la liste

# Atelier 12 : RestBuilder et Client Extension

## *12.1 RestBuilder*

Créer un module :

```
blade create -t rest-builder events-headless
```

Visualiser les 4 modules générés dans l'IDE

Définir un contrat OpenAPI

```
openapi: 3.0.1
info:
  title: Gestion des Événements (Events API)
  version: v1.0
  description: API CRUD pour la gestion des événements.
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html

components:
  schemas:
    # Définition du modèle d'objet Event
    Event:
      type: object
      properties:
        id:
          type: integer
          format: int64
          description: Identifiant unique de l'événement.
          readOnly: true # L'ID est généré par le serveur
        title:
          type: string
          description: Titre de l'événement.
          example: Conférence Groovy 2026
        dateDebut:
          type: string
          format: date-time
          description: Date et heure de l'événement.
          example: 2026-03-15T10:00:00Z
        dateFin:
          type: string
          format: date-time
          description: Date et heure de fin de l'événement.
          example: 2026-03-15T18:00:00Z


paths:
  "/events":
    # --------------------------
```

```
    # Opération : GET /events (Liste des événements)
    # ---------------------------
    get:
      tags: ["Event"]
      summary: Récupère une liste paginée de tous les événements.
      operationId: getEventsPage
      parameters:
        - in: query # Ceci définit un paramètre de requête (URL?
param=valeur)
          name: groupId
          description: Identifiant du groupe pour filtrer les
événements.
          required: false # Le paramètre est optionnel
          schema:
            type: integer
            format: int64
          style: form
          explode: true
      responses:
        200:
          description: Liste des événements réussie.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Event"
            application/xml:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Event"

    # ---------------------------
    # Opération : POST /events (Créer un nouvel événement)
    # ---------------------------
    post:
      tags: ["Event"]
      summary: Crée un nouvel événement.
      operationId: postEvent
      requestBody:
        description: Objet Event à créer.
        required: true
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Event"
          application/xml:
            schema:
              $ref: "#/components/schemas/Event"
      responses:
        201:
          description: Événement créé avec succès.
```

```
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Event"
            application/xml:
              schema:
                $ref: "#/components/schemas/Event"
        400:
          description: Requête invalide (données manquantes ou
incorrectes).
```

Générer les classes

```
./gradlew buildRest
```

Visualiser les classes générées

Ajouter la dépendance vers le service

```
 compileOnly project(":modules:events:events-api")
```

Implémentation du REST

Modifier le contenu de la classe EventResourceImpl  avec :

```
@Reference
private EventLocalService _eventLocalService;

@Override
public Page<Event> getEventsPage(Long groupId) throws Exception {

 // 2. Avoir le COUNT TOTAL (Obligatoire pour l'objet Page)
 int totalCount = _eventLocalService.getEventsCount(groupId);

 // 3. Avoir la LISTE des éléments paginés
 List<org.formation.event.model.Event> events =
 _eventLocalService.getEvents(groupId, 0, 20);

 // 4. Mapper la liste d'entités en liste de DTO
 List<Event> eventDTOs = events.stream()
 .map(this::toEventDTO)
 .collect(Collectors.toList());

 // 5. ENVELOPPER le tout dans l'objet Page
 return Page.of(eventDTOs);
}

private Event toEventDTO(org.formation.event.model.Event event) {
 return new Event() {{
 setId(event.getEventId());
 setTitle(event.getTitle());
```

```
 setDateDebut(event.getDateDebut());
 setDateFin(event.getDateFin());
 }};
}
```

Déployer

```
./gradlew deploy
```

Vérifier le déploiement

Tester

Récupérer le groupId dans la base de données

```
curl -v -H "Accept: application/json" --user "login:password"
"http://localhost:8080/o/events-headless/v1.0/events?
groupId=groupId"
```

Accéder à l'api

**http://localhost:8080/o/api**

Et afficher le swagger correspondant à notre API

## *12.2 Client extension*

Objectif

- Déployer un composant JavaScript effectuant une requête REST vers notre servie Restful

Visualiser le fichier index.js fourni et le modifier pour indiquer la bonne URL

L'uploader en tant que document et noter son URL

Dans le menu ***Application Applications personnalisées Applications distantes***
« Ajouter un élément personnalisé ».

Et remplir le formulaire comme suit :

- Nom : « Liste des Événements REST » (Nom qui apparaîtra dans la liste de widgets).
- Identifiant de l'élément HTML : ***liferay-event-list***

Attention : Ce nom doit être rigoureusement identique à celui utilisé dans le code JavaScript : customElements.define('liferay-event-list', ...).

- URL : L'url du document index.js préfixée par http://localhost:8080
- Cocher la case Utiliser l'ESM

Ensuite dans la page Présentation des événements ajouter le widget et tester son comportement

# Atelier 13 : Permissions & ServiceContext

Objectifs

- Sécuriser l'entité `Event` de bout en bout (Service, Portlet, API)
- Permettre la configuration des droits de chaque entité via l'interface Liferay.

## *13.1 Déclaration des permissions (Module API/Service)*

Mettre au point un fichier ***portlet.properties*** :

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

Modifier le fichier : `src/main/resources/resource-actions/default.xml`

Le fichier spécifie les actions supportés par le modèle de permission

```
<?xml version="1.0"?>
<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource
Action Mapping 7.4.0//EN" "http://www.liferay.com/dtd/liferay-
resource-action-mapping_7_4_0.dtd">
<resource-action-mapping>

    <model-resource>
        <model-name>org.formation.event.model.Event</model-name>
        <portlet-ref>

<portlet-name>org_formation_event_EventPortlet</portlet-name>
        </portlet-ref>
        <permissions>
            <supports>
                <action-key>VIEW</action-key>
                <action-key>UPDATE</action-key>
                <action-key>DELETE</action-key>
                <action-key>PERMISSIONS</action-key>
            </supports>
            <site-member-defaults>
                <action-key>VIEW</action-key>
            </site-member-defaults>
        </permissions>
    </model-resource>
</resource-action-mapping>
```

## *13.2 Enregistrement du ModelResourcePermission*

Créer la classe
**org.formation.event.internal.security.permission.resource.EventModelResourcePermissionRegistrar.java**

La classe déclarer le composant OSGI et déclare une référence lazy

```
@Component(immediate = true)
public class EventModelResourcePermissionRegistrar {

    private ServiceRegistration<ModelResourcePermission>
_serviceRegistration;
    @Reference(policy = ReferencePolicy.DYNAMIC, cardinality =
ReferenceCardinality.MANDATORY)
    private volatile EventLocalService _eventLocalService;

    @Reference(
            cardinality = ReferenceCardinality.OPTIONAL,
            policy = ReferencePolicy.DYNAMIC,
            target =
"(resource.name=org_formation_event_EventPortlet)"
    )
    private volatile PortletResourcePermission
_portletResourcePermission;

    @Activate
    public void activate(BundleContext bundleContext) {
        Dictionary<String, Object> properties = new
HashMapDictionary<>();
        properties.put("model.class.name", Event.class.getName());

        System.out.println("Registering ModelResourcePermission
for Event model.");
        _serviceRegistration = bundleContext.registerService(
                ModelResourcePermission.class,
                ModelResourcePermissionFactory.create(
                        Event.class, Event::getEventId,
                        _eventLocalService::getEvent,
_portletResourcePermission,
                        (modelResourcePermission, consumer) -> {
                            // Logique optionnelle ici
                        }),
                properties);
        System.out.println("ModelResourcePermission for Event
model registered."+_serviceRegistration);

    }

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


}
```

### *13.3 Sécurisation de la couche service*

Pour éviter des dépendances circulaires OSGI, on se créée une classe utilitaire qui récupère lorsque l'on en a besoin du modèle de permission :

```
public class EventPermissionHelper {
    public static ModelResourcePermission<Event>
getModelResourcePermission() {
        try {
            Bundle bundle =
FrameworkUtil.getBundle(EventLocalServiceImpl.class);
            BundleContext bundleContext =
bundle.getBundleContext();

            // 1. On définit le filtre pour cibler UNIQUEMENT
notre entité Event
            String filter =
"(model.class.name=org.formation.event.model.Event)";

            // 2. On récupère les références correspondant au
filtre
            Collection<ServiceReference<ModelResourcePermission>>
serviceReferences =

bundleContext.getServiceReferences(ModelResourcePermission.class,
filter);

            if (serviceReferences != null &&
!serviceReferences.isEmpty()) {
                // 3. On prend la première (et normalement seule)
correspondance
                ServiceReference<ModelResourcePermission> ref =
serviceReferences.iterator().next();
                return (ModelResourcePermission<Event>)
bundleContext.getService(ref);
            }
        } catch (Exception e) {
            System.err.println("Erreur lors de la récupération de
la permission Event : " + e.getMessage());
        }
        return null;
    }
}
```

Modifier EventLocalServiceImpl pour interdire l'accès en cas d'absence de droits.

On gère les cas d'ajout et de suppression

```
    public Event addEvent(long userId, long groupId, String
title, Date dateDebut, Date dateFin, ServiceContext sc) throws
PortalException {
```

```
          System.out.println("Adding event: " + title);
          PermissionChecker pc =
PermissionThreadLocal.getPermissionChecker();
          System.out.println("User ID: " + pc.getUserId());
          System.out.println("Is Omniadmin: " + pc.isOmniadmin());
          System.out.println("Group ID: " + sc.getScopeGroupId());

          boolean hasPermission = pc.hasPermission(
                    sc.getScopeGroupId(),
                    "org_formation_event_EventPortlet",
                    0, // On teste la ressource de portlet, donc
l'ID est 0 ou le scopeGroupId
                    "ADD_EVENT");
          if (!hasPermission) {
               throw new PrincipalException.MustHavePermission(
                         pc,
                         "org_formation_event_EventPortlet",
                         0,
                         "ADD_EVENT");
          }


          System.out.println("Check permission done: " + title);
          long id =
counterLocalService.increment(Event.class.getName());
          Event event = eventPersistence.create(id);

          event.setGroupId(groupId);
          event.setCompanyId(sc.getCompanyId());
          event.setUserId(userId);
          event.setTitle(title);
          event.setDateDebut(dateDebut);
          event.setDateFin(dateFin);

          event = eventPersistence.update(event);
          System.out.println("Event created with ID: " +
event.getEventId());
          System.out.println("ResouceLocalService: " +
resourceLocalService);
          System.out.println("sc.getModelPermissions() " +
sc.getModelPermissions());
          try {
               resourceLocalService.addModelResources(event, sc);
          } catch (Exception e) {
               System.err.println("ERREUR CRITIQUE : L'ajout des
ressources a échoué.");
               System.err.println("Cause : " + e.getMessage());
               // On ne relance pas l'exception pour éviter le
rollback et voir l'event en base
          }
          assetEntryLocalService.updateEntry(
                    userId,
```

```
                    sc.getScopeGroupId(),
                    Event.class.getName(),
                    event.getEventId(),
                    sc.getAssetCategoryIds(),
                    sc.getAssetTagNames()
          );
          System.out.println("After AssetEntryLocalService
updateEntry");
          return event;
     }

     // CAS 2 : Action avec ID (Model Resource)
     public void deleteEvent(long eventId, ServiceContext
serviceContext) throws PortalException {
System.out.println("Deleting event ID: " + eventId);
          PermissionChecker pc =
PermissionThreadLocal.getPermissionChecker();
          boolean hasDeletePermission = pc.hasPermission(
                    serviceContext.getScopeGroupId(),
                    "org.formation.event.model.Event",
                    eventId,
                    ActionKeys.DELETE
          );
          System.out.println("L'utilisateur a-t-il le droit DELETE
? " + hasDeletePermission);
          ModelResourcePermission<Event>  modelResourcePermission
= EventPermissionHelper.getModelResourcePermission();
System.out.println("ModelResourcePermission " +
modelResourcePermission.getModelName());
          // On vérifie si l'utilisateur a le droit de supprimer
CET événement précis
          modelResourcePermission.check(
                    PermissionThreadLocal.getPermissionChecker(),
                    eventId,
                    ActionKeys.DELETE);

          deleteEvent(eventId);
          System.out.println("Event deleted ID: " + eventId);
     }
     public List<Event> getEvents(long groupId, int start, int
end) {
          return eventPersistence.findByGroupId(groupId,  start,
end);
     }

     public int getEventsCount(long groupId) {
          return eventPersistence.countByGroupId(groupId);
     }

}
```

Regénérer le service et déployer

### *13.3 Adaptation de l'interface (Module Web)*

**Fichier :** `EventListMVCRenderCommand`

**On positionne dans le mode le modèle de permission de Event**

```
@Reference(
        target =
"(model.class.name=org.formation.event.model.Event)",
        unbind = "-"
)
private ModelResourcePermission<Event>
_eventModelResourcePermission;
```

**….**

```
renderRequest.setAttribute("eventModelResourcePermission",
_eventModelResourcePermission);
```

**Fichier :** `view.jsp`
Donner la Possibilité de configurer les permissions par entité

```
<%-- Import du moteur de permission --%>
<%
ModelResourcePermission<Event> eventModelResourcePermission =
    (ModelResourcePermission<Event>)
request.getAttribute("eventModelResourcePermission");
%>

<liferay-ui:search-container-row
className="org.formation.event.model.Event" modelVar="event">
    <liferay-ui:search-container-column-text property="title" />

    <liferay-ui:search-container-column-text name="actions">
        <%-- Bouton Modifier conditionnel --%>
        <c:if test="<%=
eventModelResourcePermission.contains(permissionChecker,
event.getEventId(), ActionKeys.DELETE) %>">
            <portlet:renderURL var="deleteURL">
                <portlet:param name="mvcPath"
value="/delete_event.jsp" />
                <portlet:param name="eventId" value="<%=
String.valueOf(event.getEventId()) %>" />
            </portlet:renderURL>
            <a href="${deleteURL}">Modifier</a>
        </c:if>

        <%-- Bouton de configuration des permissions (Délégation)
```

```
--%>
        <c:if test="<%=
eventModelResourcePermission.contains(permissionChecker,
event.getEventId(), ActionKeys.PERMISSIONS) %>">
            <liferay-security:permissionsURL
                modelResource="<%= Event.class.getName() %>"
                modelResourceDescription="<%= event.getTitle() %>"
                resourcePrimKey="<%=
String.valueOf(event.getEventId()) %>"
                var="permissionsURL"
            />
            <a href="${permissionsURL}">Permissions</a>
        </c:if>
    </liferay-ui:search-container-column-text>
</liferay-ui:search-container-row>
```

Déployer le tout

### *13.4 Test fonctionnel*

- En tant qu'Admin : Allez sur le widget, cliquez sur les points de suspension d'un événement -> "Permissions". Donnez le droit "Update" au rôle "Site Member".
- En tant qu'Utilisateur (Site Member) : Connectez-vous. Le bouton "Modifier" doit apparaître.
- Retrait des droits : Enlevez la permission. Le bouton doit disparaître de la JSP ET si l'utilisateur tente de forcer l'URL, le service doit lever une `PrincipalException`.

# Atelier 14 : Asset, Thème, OSGI

## *14.1 Asset Framework*

Objectifs

- Associer des catégories et tags à une entité personnalisée (`Event`)
- Utiliser `ServiceContext` pour alimenter Asset Framework
- Afficher les tags/catégories dans l'interface
- Tester l'intégration avec Asset Publisher

Ajouter les sélecteurs de tags / catégories dans la JSP

Dans `edit.jsp` , ajouter les sélecteurs dans le formaulaire

```
<div class="my-3">
    <liferay-asset:asset-categories-selector
            className="<%= Event.class.getName() %>"
            classPK="<%= 0 %>"
    />
    <liferay-asset:asset-tags-selector
            className="<%= Event.class.getName() %>"
            classPK="<%= 0 %>"
    />
</div>
```

Récupérer tags & catégories dans l'ActionCommand

Dans l'ActionCommand, les catégories et les tags sont dans le servicecontext, vous pouvez ajouter ce code pour inspecter le ServiceContext

```
/ 1. Vérification des Tags
        String[] tags = serviceContext.getAssetTagNames();
        System.out.println("Tags trouvés : " + (tags != null ? String.join(", ",
tags) : "aucun"));

// 2. Vérification des Catégories
        long[] categoryIds = serviceContext.getAssetCategoryIds();
        if (categoryIds != null) {
            for (long id : categoryIds) {
                System.out.println("ID Catégorie sélectionnée : " + id);
            }
        } else {
            System.out.println("Catégories : aucune");
        }

// 3. Vérification du Scope (très important pour l'Asset Framework)
        System.out.println("Scope GroupId : " +
serviceContext.getScopeGroupId());
        System.out.println("---------------------------------");
```

Appeler AssetEntryLocalService dans la couche service

Normalement AssetEntryLocalService est déjà injecté via notre service.xml

Ajouter l’appel :

```
assetEntryLocalService.updateEntry(
    userId,
    serviceContext.getScopeGroupId(),
    Event.class.getName(),
    event.getEventId(),
    serviceContext.getAssetCategoryIds(),
    serviceContext.getAssetTagNames()
);
```

Vérifcation

En base :

```
SELECT * FROM AssetEntry WHERE classNameId = (SELECT classNameId
FROM ClassName_ WHERE value = 'org.formation.event.model.Event')
```

Dans `view.jsp, ajouter une colonne` :

```
<liferay-ui:search-container-column-text
          name="Tags/Catégories">
<p>Tags :
     <liferay-asset:asset-tags-summary
               className="<%= Event.class.getName() %>"
               classPK="<%= event.getEventId() %>" />
</p>
<p>Catégories :
     <liferay-asset:asset-categories-summary
               className="<%= Event.class.getName() %>"
               classPK="<%= event.getEventId() %>" />
</p>
</liferay-ui:search-container-column-text>
```

## *14.2 Thème*

Objectifs

- Créer un thème minimaliste
- Modifier la structure via Freemarker
- Ajouter du style personnalisé via SCSS
- Tester le thème dans le portail

Générer un thème

Dans le Liferay Workspace :

```
blade create -t theme custom-theme
```

Importer dans l’IDE.

Modifier le SCSS

Dans `src/css/_custom.scss` :

```
body {
   background-color: #f5f7fa;
}

#header {
   background-color: #004b8d;
   color: white;
}
```

Modifier le template principal

Dans `src/templates/portal_normal.ftl` :

```
<#include init />


<div class="alert alert-info text-center">
    Thème personnalisé chargé avec succès
</div><!DOCTYPE html>
<html dir="ltr" lang="${w3c_language_id}">
<head>
    <title>${the_title}</title>
    <@liferay_util["include"] page=top_head_include />
</head>
<body class="${css_class}">
    <@liferay_util["include"] page=body_top_include />

    <div class="container-fluid" id="wrapper">
        <header id="banner" role="banner">
            <h1 class="site-title">${site_name}</h1>
        </header>

        <#-- AJOUT DU TP ICI -->
        <div class="alert alert-info text-center">
            Thème personnalisé chargé avec succès
        </div>

        <section id="content">
            <#if selectable>
                <@liferay_util["include"] page=content_include />
            <#else>
                ${portlet_display.recycle()}
                ${portlet_display.setTitle(the_title)}
                <@liferay_util["include"] page=content_include />
            </#if>
```

```
        </section>

        <footer id="footer" role="contentinfo">
            <p>Mon Nouveau Thème 2025</p>
        </footer>
    </div>

    <@liferay_util["include"] page=body_bottom_include />
    <@liferay_util["include"] page=bottom_include />
</body>
</html>
```

Déployer

```
./gradlew deploy
```

Appliquer le thème à un site.

# Atelier 15 : Personnalisation OSGI

## *15.1 Personnalisation via Fragment Module (Override JSP)*

Objectifs

- Apprendre la méthode moderne pour remplacer une JSP native
- Comprendre le fonctionnement des OSGi Fragment Modules
- Modifier l'IHM de Liferay sans EXT ni Hooks

Créer un fragment

Retrouver la version du bundle com.liferay/login.web via Gogo Shell :

lb -s | grep login.web

Utiliser la version dans la commande

```
blade create -t fragment -h com.liferay.login.web -H $version
login-fragment
```

Cela crée un module dépendant du bundle ***com.liferay.login.web.***

Identifier la JSP à remplacer

Exemple : remplacer le texte dans `login.jsp` (emplacement réel dépend de la version).

```
unzip -p ./osgi/portal/com.liferay.login.web.jar
META-INF/resources/login.jsp > ~/Desktop/login_original.jsp
```

Accepter l'arborescence proposée :

```
src/main/resources/
   META-INF/resources/
      login.jsp
```

Modifier la JSP

Ajouter un message :

```
<div class="alert alert-warning">
   Message personnalisé : connexion au portail Liferay
</div>
```

Déployer

```
./gradlew deploy
```

Aller sur `/c/portal/login` pour valider.

Validation

- Le message personnalisé apparaît au-dessus du formulaire de connexion.

## *15.2 Personnalisation via Service Wrapper OSGi*

Objectifs

- Surcharger un service existant de Liferay
- Intercepter une opération métier
- Ajouter des règles ou logs personnalisés

Créer un module wrapper

```
blade create -t service-wrapper -s
com.liferay.portal.kernel.service.UserLocalService user-wrapper
```

Modifier la classe générée

Dans `UserLocalServiceWrapper` :

```
@Component(
    property = {
    },
    service = ServiceWrapper.class
)
public class UserWrapper extends UserLocalServiceWrapper {

    public UserWrapper() {
        super(null);
    }

    @Override
    public User addUser(User user) {
        System.out.println(">>> [WRAPPER] Interception de l'ajout
de l'utilisateur : " + user.getFullName());
        return super.addUser(user);
    }
}
```

Déployer

Créer un utilisateur → observer les logs serveur.

Validation

- Le log avant/après création apparaît correctement.
- Aucune régression fonctionnelle.

## *15.3 : Model Listener OSGi*

Objectifs

- Réagir aux événements CRUD sur une entité
- Implémenter la logique d’audit ou de synchronisation

- Découvrir le hook moderne remplaçant les “Value Object Listeners”

Créer un module listener

```
blade create -t api -p org.formation.events.listener events-listener
```

Ajouter un ModelListener

```
@Component(
    service = ModelListener.class
)
public class EventModelListener extends BaseModelListener<Event> {

    @Override
    public void onAfterCreate(Event event) {
        System.out.println("Événement créé : " + event.getName());
    }

    @Override
    public void onAfterRemove(Event event) {
        System.out.println("Événement supprimé : " + event.getName());
    }
}
```

Déployer et tester

Créer et supprimer un Event → vérifier les logs.

Validation

- Les messages apparaissent lors de la création et suppression.