DOI:**10.33099/2311-7249/2020-38-2-71-80**

УДК: 004.4`2

Юрій Васильович Кравченко *(доктор технічних наук, професор)*[1]
Максим Георгійович Тищенко *(кандидат технічних наук)*[1]
Олександр Олександрович Шапран[1]
Євген Олександрович Судніков[1]
Валентин Геннадійович Твардовський[2]

[1]***Національний університет оборони України імені Івана Черняховського, Київ, Україна***
[2]***Київський національний університет імені Тараса Шевченка, Київ, Україна***

МЕТОДИКА РОЗРОБКИ WEB-ДОДАТКУ НА ОСНОВІ ПОРТАЛУ LIFERAY

Сервіси, доступні через веб-інтерфейс, стали невід'ємною складовою частиною сучасного світу. У зв'язку зі зростанням числа Web-додатків, їх підтримка і адміністрування стає все більш складним завданням. Таким чином, рано чи пізно практично перед будь-якою організацією постає завдання інтеграції цих сервісів.

Оптимальним варіантом вирішення описаної вище проблеми є розгортання корпоративного порталу, який забезпечить централізований доступ до всіх веб-сервісів організації.

В переважній більшості, розробники Web-додатків для вирішення зазначених задач використовують корпоративний портал Liferay, який є потужною та сучасною платформою з відкритим вихідним кодом. На теперешній час є одним з найбільш популярних рішень при побудові корпоративних порталів.

Але у розробників виникає проблема із наявністю у вільному доступі матеріалів та документації, яка б у повній мірі розкривала весь можливий спектр реалізації функціоналу платформи.

Отже, існує необхідність в детальному аналізі функціоналу та прикладів реалізації на основі порталу Liferay для розробки Web-додатків.

В статті проведено аналіз базового принципу розробки Web-додатків, його реалізацію в досліджуваному корпоративному порталі Liferay, а також в доступних для розробників модулях. Розкрито функціонал інструменту для розробки проектів Liferay Digital Experience Platform (DXP).

Проаналізована базова структура основних компонентів порталу для розробки корпоративних задач Liferay і на основі цієї інформації проаналізовані ефективні приклади реалізації цих компонентів, які можна застосовувати для створення структури Web-додатків будь-якого типажу і з будь-якими задачами.

Ключові слова*: Liferay; Web-додаток; Model View Controller; Liferay Digital Experience Platform.*

Вступ

У сучасному світі практично у кожної людини є доступ до Інтернету і велика частина з них користується ним майже кожен день. Веб-програмування швидко стає привабливою і високооплачуваною сферою діяльності в нашому світі. І одна з найбільш неприємних проблем, з якими зіштовхуються веб-програмісти – це постійне повторення написання базового коду при створенні нових Web-додатків.

Кожен новий проект має приблизно однакову початкову структуру: розробник пише новий набір таблиць баз даних, новий API, новий набір класів CSS та HTML, новий набір функцій JavaScript. Переважно всі додатки чи сайти мають опцію реєстрації на сайті, а також вхід до свого профілю та вихід з нього. Коли розробники дотримуються кращих практик реалізації такого функціоналу – він, переважно, співпадає з написаним кодом у всіх сайтах та додатках. Фактично відбувається повторюваність одних і тих самих дій, що може призвести до типових помилок, а програмування на те й існує, щоб автоматизувати процеси, в яких людина може створити помилку. Існує необхідність щоб існувала платформа, яка б надавала базовий набір функцій, що дадуть розробнику для початкового розроблення усього того функціоналу і, відповідно, коду, що повторюється. На сьогодні існують такі платформи: IBM Websphere, SharePoint, Drupal, Liferay, Adobe Experience Manager, Magento, Jahia [2, 4, 5, 7].

Постановка проблеми. Враховуючи базову структуру основних компонентів порталу для розробки корпоративних задач Liferay, необхідно розробити ефективні приклади реалізацій цих компонентів, які можна застосовувати для створення Web-додатків. Ці реалізації мають бути без спеціалізованого налаштування і підходити для будь-якого типу Web-додатку. Потрібно розробити реалізацію загального варіанту

контролера типу Model View Controller (MVC) Portlet, а також окремі реалізації для спеціалізованих контролерів, таких як MVC команди дії, MVC команди візуалізації, MVC команди ресурсів, які мають опрацьовувати конкретні бізнес процеси в додатках. А для взаємодії будь-якого додатку з базою даних, не прив'язуючись до конкретного реалізації базою даних, необхідно розробити реалізацію такого компоненту, як Service Builder. Використовуючи всі ці реалізації основних компонентів Liferay Portal, розробник зможе розуміти принципи та ідеї, закладені в спеціальну реалізацію шаблону MVC від Lifeary, і застосовувати ці компоненти в будь-якому Web-додатку, а також використовувати та модернізувати вже існуючі в Liferay DXP загальні програми.

Аналіз останніх досліджень і публікацій. Матеріали для початку роботи з цією платформою обмежуються лише офіційною документацією Liferay Portal [1] та декількома посібниками Йонаса Юань [15,16].

У роботі Саміра Бхатт [17] розглянуті архітектурні варіанти та найкращі практики. Зосереджено увагу про наслідки різних архітектурних варіантів, про те, як налаштувати портал Liferay для роботи в кластерному середовищі. Обговорюються різні варіанти, доступні в конфігурації кластера. Також зазначені різні варіанти конфігурації різних компонентів, які доступні для підвищення продуктивності.

Нажаль, всі ці матеріали доступні лише англійською мовою та не в повному обсязі дають змогу зрозуміти архітектуру побудови Web-додатків.

Отже, ця стаття може допомогти українським розробникам швидше навчитись використовувати цей потужний інструмент для розробки власних проектів та для проведення досліджень при створені Web-додатків на базі Liferay Portal або в разі необхідності доопрацювати або модернізувати вже створені проекти, що базуються на цьому корпоративному порталі

Метою статті є проведення аналізу основних інструментів для розробки в корпоративному порталі Liferay та розкриття функціональності компонентів MVC Portlet, MVC Action Command, MVC Render Command, MVC Resource Command та Service Builder, що презентуватимуть варіант ефективного використання порталу Liferay для розробки сайтів.

Виклад основного матеріалу дослідження

Портал Liferay – платформа для розробки програмного забезпечення Open Source Enterprise, заснована на сучасній технології J2EE. Liferay простіший за WebSphere [5] і гнучкіший, ніж SharePoint [6]. Це як динамічна, так і високо масштабна платформа, за допомогою якої створюються красиві інтерактивні веб-сайти та портали підприємств. Liferay DXP містить кілька функціональних підрозділів під назвою портлет, які пропонують широку підтримку багатьох мов програмування, таких як Java, C++, PHP, .NET та багато інших. Цих причин достатньо, щоб розглянути можливість впровадження порталів Liferay у своїх організаціях, проте є багато інших переконливих, таких як:

1. Повне рішення потреб підприємств. Є два видання Liferay. Видання Community та видання Enterprise. Видання Enterprise надає всю підтримку управління організацією для всіх заходів, пов'язаних із розробкою веб-порталів, оскільки вся програма може бути розроблена без написання єдиного рядка коду. У виданні Community Liferay є багато портлетів та доповнень, які працюють на стимулювання між різними розробками порталів Enterprise.

2. Ефективність. Ви можете розробити свій корпоративний портал, не сплачуючи ні копійки. Liferay – це рішення з низьких витрат на розробку порталу, оскільки воно не передбачає витрат на ліцензування як таких. Однак, можливо, вам доведеться заплатити за Enterprise видання, щоб мати можливість використовувати всі його готові реалізації.

3. Проста реалізація. Liferay базується на платформі Java. Таким чином, реалізувати свою програму легко, використовуючи безліч ресурсів, які вже доступні на ринку як в Інтернеті, так і в автономному режимі.

4. Гнучкість розвитку. Liferay може інтегруватися практично з усіма застарілими системами ERP та CMS. Він також передбачає інтеграцію із сучасними та старими технологіями, такими як JBoss, Spring, Hibernate та інші бази даних [11]. Це дозволяє ефективно розробникам змінювати конфігурації та параметри системи для використання відповідно до ваших вимог.

5. Сумісність з UI/UX. Він може працювати з CSS, XHTML, HTML5 та низкою інших технологій проектування. Крім того, динамічна функція "Drag and Drop" робить цікавою для розробників та користувачів переміщувати різні об'єкти на своїх порталах.

6. Персоналізований досвід. Liferay дозволяє користувачам персоналізувати сторінки своїх веб-сайтів та веб-порталів. Це дозволяє надійній системі управління вмістом підприємства легко додавати (редагувати або видаляти) та змінювати вміст на порталах та веб-сайтах з можливістю зробити його загальнодоступним або приватним, що робить його найкращою платформою цифрового досвіду.

7. Упорядкований робочий процес. Liferay надає унікальні API робочого процесу для кращого користувацького досвіду. Використовуючи ці інструменти, користувач може заощадити час та уникнути небезпеки розвитку. Liferay дозволяє користувачам створювати та застосовувати власний робочий процес із спеціальними сутностями.

8. Вихідні модулі. Інструмент Liferay постачається з більш ніж 60 готовими до використання портлетами. Ці портлети завжди в режимі перетягування та розгортання. І той же портлет може використовуватися для адміністрування різних веб-сайтів одним користувачем.

Створення Liferay MVC Portlet. Що для цього знадобиться:

модуль, який публікує компонент портлета з необхідними властивостями;

код контролера для обробки запиту та відповіді;

JSP для реалізації шару перегляду.

Попутно потрібно знати, як викликати сервіси зі свого контролера та як передавати інформацію з рівня перегляду в контролер. Важливо не забувати, що для реалізації портлету Liferay MVC наявні два шляхи. Якщо додаток є невеликим за функціональністю, який не буде важким для логіки контролера (можливо, всього лише кілька способів дій), то можна помістити весь код свого контролера в клас – Portlet. Якщо є більш складні потреби (безліч дій, складна логіка візуалізації для реалізації або, можливо, навіть якийсь код обслуговування сервісу), краще подумати про розбиття контролера на класи MVC Action Command, MVC Render Command та MVC Resource Command [12-14].

MVC Action Command (команда дії). Liferay MVC фреймворк дозволяє розділити методи дії вашого портлету на окремі класи. Це може бути дуже корисно для портлетів, які мають багато дій. Кожна URL-адреса дії в JSP-програмах вашого портлету тоді викликає відповідний клас дій, коли це необхідно.

Спочатку використовується тег <portlet:actionURL>, щоб створити URL-адресу дії у своєму JSP. Наприклад, дія редагування запису в щоденнику в додатку “Блоги Liferay” визначається у файлі edit_entry.jsp таким чином:

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

Коли запускається URL-адреса дії, відповідний клас дій обробляє дію. Потрібно реалізувати дію, створивши клас, який реалізує інтерфейс MVCActionCommand. Щоб уникнути написання кодів кодової панелі, клас *MVCActionCommand повинен поширити клас BaseMVCActionCommand замість того, щоб безпосередньо застосовувати MVCActionCommand. Клас BaseMVCActionCommand вже реалізує MVCActionCommand і надає багато корисних реалізацій методів. Надавати ім’я класу *MVCActionCommand врахувуючи дію, яку він виконує, є хорошою практикою. Наприклад, якщо дія редагує якийсь запис, можна назвати її клас EditEntryMVCActionCommand.

MVC Render Command (команда візуалізації). Для використання команд візуалізації MVC потрібні такі речі:

впровадження інтерфейсу MVCRenderCommand;

URL-адреса відображення портлету у шарі представлення;

компонент, який публікує послугу MVCRenderCommand з двома властивостями.

MVC Resource Command (команда ресурсів). Використовуючи Liferay MVC фреймворк, можна створювати URL-адреси ресурсів у JSP для отримання зображень, XML або будь-якого іншого виду ресурсу з екземпляра Liferay. Потім URL-адреса ресурсу викликає відповідний командний клас ресурсу MVC (*MVCResourceCommand), який обробляє запит та відповідь на ресурс.

Спочатку використовується тег <portlet:resourceURL>, щоб створити URL-адресу ресурсу в JSP. Наприклад, файл /login-web/src/main/resources/META-INF/resources/navigation/create_account.jsp портлета для входу визначає наступну URL-адресу ресурсу для отримання образу CAPTCHA під час створення облікового запису:

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

Коли запускається URL-адреса ресурсу, клас *MVCResourceCommand, що відповідає, обробляє запит та відповідає. Можна створити цей клас, застосувавши інтерфейс MVCResourceCommand або розширивши клас BaseMVCResourceCommand. Останнє може заощадити час, оскільки воно вже реалізує MVCResourceCommand.

Крім того, непогано назвати клас *MVCResourceCommand відповідно ресурсу, яким він обробляє, і додати як суфікс до MVCResourceCommand. Наприклад, командний клас ресурсу, що відповідає попередній URL-адресі ресурсу CAPTCHA в портлеті входу, є CaptchaMVCResourceCommand. У додатку з декількома класами команд MVC це допоможе розрізнити їх.

Service Builder – компонент для взаємодії з БД. Першим кроком у використанні Service Builder [8] є визначення модельних класів та їх атрибутів у файлі service.xml. Розташування цього файлу зазвичай знаходиться в кореневій папці *-сервісного модуля, хоча розробник може налаштувати інструмент збирання, щоб розпізнати його з інших каталогів. У термінології Service Builder класи моделей називаються об’єктами. Наприклад, програма “Закладки” має два об’єкти: BookmarksEntry та BookmarksFolder. Вимоги до кожного з цих об’єктів визначені в службі модуля закладок service.xml, вказаному в елементах <column />.

Після того як Service Builder прочитає файл service.xml, розробник може визначити свої сутності. Liferay Developer Studio дозволяє легко визначити об’єкти у файлі service.xml для програми. Щоб визначити спеціальну сутність, потрібно виконати ряд дій.

Визначення інформації про глобальну службу. Глобальна інформація сервісу стосується

всіх її суб'єктів, тому це хороше місце для початку. У Liferay Developer Studio необхідно вибрати вузол Service Builder у верхньому лівому куті режиму огляду файлу service.xml. На рис.1 представлена форма Service Builder, в якій можна вводити глобальну інформацію сервісу. Як видно нижче з рис.1, поля містять шляхи пакета сервісу, автора та простору імен.

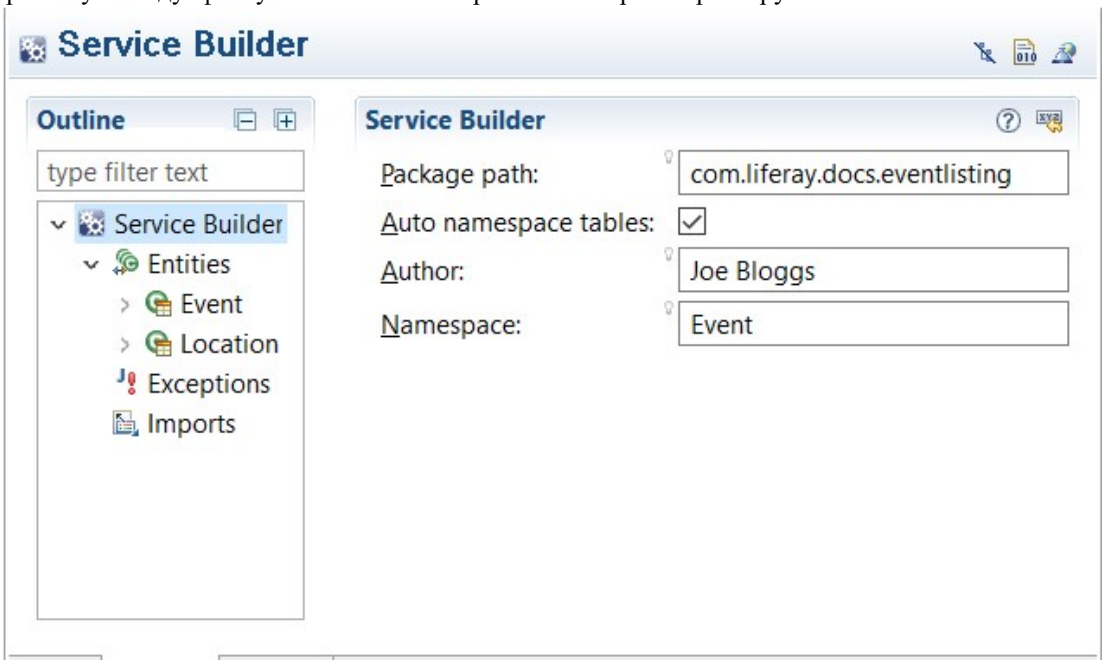

Рис. 1. Форма Service Builder з вигаданої програми "Список даних про події", "service.xml"

Шлях пакета вказує пакет, в якому створюються сервісні та персистентні класи. Шлях пакета, визначений вище, гарантує, що класи обслуговування генеруються в пакеті com.liferay.docs.eventlisting в модулі *-api. Класи персистентності генеруються в однойменному пакеті *- сервісного модуля. Наприклад, розробник може подивитися в модулях закладок - api та модулях служб закладок, щоб переглянути приклад того, як вони автоматично створюються для.

Service Builder використовує простір імен служб для іменування таблиць баз даних, які він створює для послуги. Наприклад, подія може служити простором імен для сервісу портлету Event Listing.

<namespace>Event</namespace>

Service Builder використовує простір імен у таких сценаріях SQL, які він генерує у папці src/main/ресурси/META-INF/sql:

indexes.sql
sequences.sql
tables.sql

Варто зауважити, що місце папки для зберігання створених сценаріїв SQL можна налаштувати. Наприклад, якщо використовується Ant, розробник може налаштувати аргумент у своєму build.xml, подібному до цього:

<arg value="service.sql.dir=${basedir}/../sql"/>

Якщо розробник використовує Gradle, він може визначити налаштування sqlDir у файлі build.gradle проекту або у файлі Maven pom.xml так само, як у наведених нижче прикладах застосовується налаштування databaseNameMaxLength. Liferay DXP використовує ці сценарії для створення таблиць баз даних для всіх об'єктів, визначених у файлі service.xml. Service Builder попередньо додає простір імен до імен таблиці бази даних. Оскільки значенням простору імен вище є Event, імена таблиць баз даних, створених для об'єктів, починаються з Event_ як їх префікса. Простір імен для кожного проекту Service Builder повинен бути унікальним. Окремі плагіни повинні використовувати окремі простори імен і не повинні використовувати простір імен, які вже використовуються Liferay (наприклад, Users або Groups).

Gradle build.gradle

```
buildService {
    ...
    databaseNameMaxLength = 64
    ...
}
```

Maven pom.xml

```
<configuration>
    ...
<databaseNameMaxLength>64</databaseNameMaxLength>

</configuration>
```

Як останній фрагмент глобальної інформації, можна вести ім'я як автора служби у файл service.xml. Service Builder додає анотації @author із вказаним іменем до всіх створених Java-класів та інтерфейсів. Потрібно зберегти файл service.xml, щоб зберегти додану інформацію. Потім розробник може додати об'єкти для подій та місцеположень сервісу.

Визначення сутностей сервісу. Сутності – це серце і душа служби. Сутності представляють

карту між об'єктами моделі на Java та полями і таблицями у базі даних. Після того, як сутності визначені, Service Builder обробляє відображення автоматично, надаючи можливість приймати об'єкти Java та зберігати їх. Для програми Закладки створено два об'єкти відповідно до його служби .xml – один для записів закладок та один для папок закладок. Ось підсумок інформації, що використовується для об'єкта BookmarksEntry:

назва: Закладки;

місцева служба: так;

віддалене обслуговування: так.

Ось що було використано для сутності BookmarkFolder:

назва: Закладки;

місцева служба: так;

віддалене обслуговування: так.

Щоб створити сутності за допомогою програми Liferay Developer Studio, потрібно вибрати вузол Entities під вузлом Builder служби в контурі зліва від редактора service.xml у режимі огляду. У головній частині перегляду важливо зауважити, що таблиця Entities порожня. Необхідно створити об'єкт, натиснувши на значок Додати об'єкт (+) праворуч від таблиці. Далі потрібно ввести ім'я організації та, якщо розробник хоче генерувати локальні та віддалені послуги для цієї організації можна додавати стільки об'єктів, скільки потрібно. Додавання об'єктів сервісу легко за допомогою режим *Overview* програми Liferay Developer Studio у файлі `service.xml', який зображено на рис. 2.

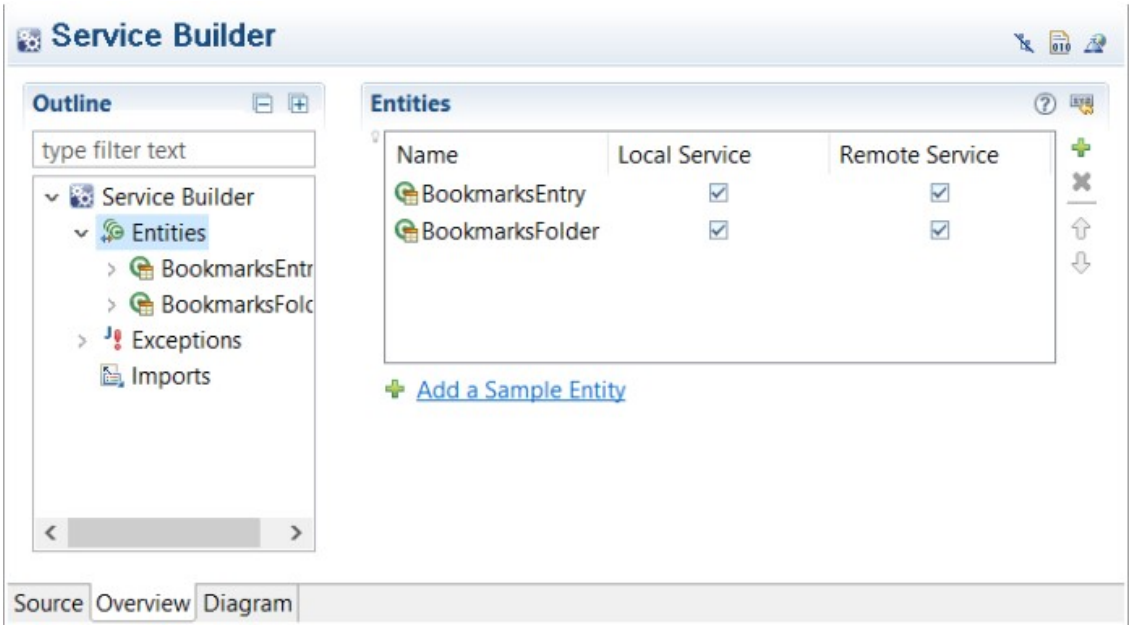

Рис. 2. Режим *Overview* програми Liferay Developer Studio у файлі `service.xml'

Ім'я організації використовується для імені таблиці бази даних для збереження екземплярів сутності. Фактична назва таблиці бази даних має префікс простору імен; у прикладі Закладки створюється одна таблиця бази даних з назвою Bookmarks_BookmarksEntry та інша назва Bookmarks_BookmarksFolder.

Встановлення атрибуту локального сервісу на істинний інструктор Builder Service генерує локальні інтерфейси для служб суб'єкта господарювання. Значення за замовчуванням для локального сервісу – false. Локальні сервіси можна викликати лише з сервера Liferay, на якому вони розміщені. Тому, якщо ваша програма буде розгорнута до Liferay, сервіс буде локальний з точки зору вашого сервера Liferay. Встановлення атрибута віддаленого сервісу на істинний інструктор Builder Service створює віддалені інтерфейси для сервісу. Значення за замовчуванням для віддаленої служби є true. Розробник може створити повністю функціональний додаток без генерації віддалених служб. У такому випадку можна встановити локальну службу на true, а віддалену службу на false для своїх організацій. Якщо необхідно ввімкнути віддалений доступ до служб свого додатка, слід встановити значення true як для місцевої, так і для віддаленої служби.

Визначення стовпців (атрибутів) для кожної організації. Кожна сутність описується своїми стовпцями, які представляють атрибути суб'єкта. Ці атрибути відображаються з одного боку на полях таблиці, а з іншого – на атрибути об'єкта Java. Щоб додати атрибути для сутності, потрібно перейти до її стовпців у режимі огляду контуру файлу service.xml. З контуру розгорнути вузол Entities і розгорнути вузол сутності. Потім вибрати вузол Стовпці. Liferay Developer Studio відображає таблицю стовпців організації.

Service Builder створює поле бази даних для кожного стовпця, який ви додаєте у файл service.xml. Він відображає тип поля бази даних, відповідний типу Java, визначеному для кожного стовпця, і це робить для всіх баз даних, що підтримують Liferay. Після запуску Service Builder він генерує конфігурацію Hibernate, яка обробляє

об'єктно-реляційне відображення. Service Builder автоматично генерує методи getter/setter у класі моделі для цих атрибутів. Ім'я стовпця вказує ім'я, яке використовується у геттерах та сетерах, створених для поля Java-сутності організації. Тип стовпця вказує тип Java для цього поля для сутності. Якщо значення атрибута Основного (тобто первинного ключа) стовпця встановлено на істинне, то стовпець стає частиною первинного ключа для сутності.

Первинний ключ сутності – це унікальний ідентифікатор для сутності. Якщо лише для одного стовпця встановлено значення true, то цей стовпець представляє весь первинний ключ для сутності. Однак можна використовувати кілька стовпців як основний ключ для сутності. У цьому випадку комбінація стовпців складає складний первинний ключ для сутності. Аналогічно тому, як використовувалася таблиця форм для додавання об'єктів, потрібно додати стовпці атрибутів для кожної сутності.

Кожен атрибут можна створити, натиснувши на значок Додати (+). Потім заповнити ім'я атрибута, вибрати його тип та вказати, чи це первинний ключ для сутності. Поки курсор знаходиться у полі Тип стовпця, з'являється значок опції. Потрібно натиснути на цей значок, щоб вибрати відповідний тип стовпця. І так потрібно створити стовпець для кожного атрибуту сутності чи сутностей.

До того ж, можна додати стовпці, щоб допомогти перевірити свої сутності. Наприклад, можна створити стовпець з назвою createDate типу Date, щоб відзначити дату створення екземпляра сутності. І додати стовпець з назвою modifiedDate типу Date, щоб відстежувати останній раз, коли екземпляр сутності був змінений. Приклад такої сутності з додатковими стовпцями наведено в таблиці 1.

Таблиця 1.

Примітивний приклад сутності користувача з первинним ключем та стовпцями, що відповідають за аудит сутності

Назва	Тип	Первинний ключ
userId	long	+
createDate	Date	-
modifiedDate	Date	-

Визначення відносин між сутностями. Як було сказано раніше, кожна закладка повинна мати папку. Отже, кожен об'єкт BookmarksEntry повинен мати відношення до сутності BookmarksFolder. Режим діаграм програми Liferay Developer Studio для service.xml спрощує відповідні об'єкти. Спочатку потрібно вибрати режим діаграми для файлу service.xml. Потім виберіть параметр "Зв'язок" у розділі "З'єднання" на палітрі правої сторони екрана. Цей інструмент взаємозв'язку допомагає скласти відносини між сутностями на діаграмі. Далі вибрати свою першу сутність і пересунути курсор на об'єкт, з яким потрібно пов'язати його. Liferay Developer Studio малює пунктирною лінією від вибраної сутності до курсору. Далі необхідно натиснути на другу сутність, щоб завершити малювання відносин. Liferay IDE перетворює пунктирну лінію в суцільну лінію, стрілка вказує на другу сутність. Далі потрібно зберегти файл service.xml. Це можна побачити на рис. 3.

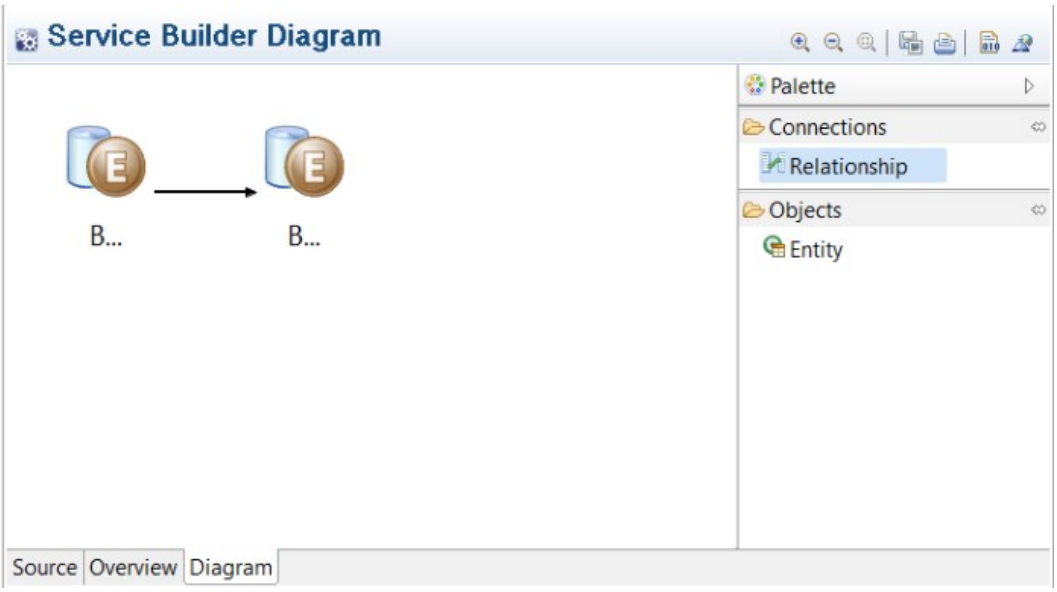

Рис. 3. Режим *Diagram* програми Liferay Developer Studio для `service.xml'

Визначення сортування екземплярів Service Entity Liferay дає змогу вказати типовий порядок об'єктів у файлі service.xml. Припустимо, є необхідність повернути сутності BookmarksEntry в алфавітному порядку за назвою. Вказати ці значення за замовчуванням легко за допомогою Liferay Developer Studio. Потрібно повернутися до режиму огляду в редакторі файлу service.xml. Потім вибрати вузол "Порядок" під вузлом сутності в контурі зліва від зображення. IDE

відображає форму для сортування обраної сутності.

Потрібно встановити прапорець “Вказати порядок”, щоб відобразити форму для встановлення порядку. Далі створити стовпець замовлення, натиснувши значок Додати (+)

Визначення методів пошуку екземплярів сутностей. Методи Finder витягують об’єкти сутності із бази даних на основі заданих параметрів. Service Builder генерує кілька методів на основі кожного пошуку, який розробник створює для сутності. Він створює методи для отримання, пошуку, видалення та підрахунку екземплярів сутності на основі параметрів пошуку. Для багатьох застосунків важливо мати можливість знаходити його об’єкти на одному сайті. Можна вказати ці пошукові методи, використовуючи режим огляду служби Liferay Developer Studio в service.xml. Далі потрібно вибрати вузол Finders під вузлом сутності в праворуч від таблиці. Потім ввести назву стовпця (наприклад, ім’я, дату тощо), яку потрібно використовувати для сортування об’єкта. Натиснути піктограму “Огляд” та вибрати опцію “Висхідний” або “Нисхідний”. Це впорядковує сутності у порядку зростання чи зменшення. контуру в лівій частині екрану. IDE відображає порожню таблицю Finders у головній частині подання. Тут створюється новий пошук, натиснувши значок Додати (+) праворуч від таблиці. Після чого вказуємо методу ім’я та тип повернення. Під новим вузлом пошуку IDE створив вузол стовпців Finder. Розробнику потрібно вибрати вузол “Стовпці шукача”, щоб вказати стовпці для параметрів пошуку.

Потім створити новий стовпець пошуку, натиснувши піктограму Додати та вказавши його ім’я. Можна вказати кілька параметрів (стовпців) для пошуку. Приклад створення методів для пошуку можна побачити на рис. 4.

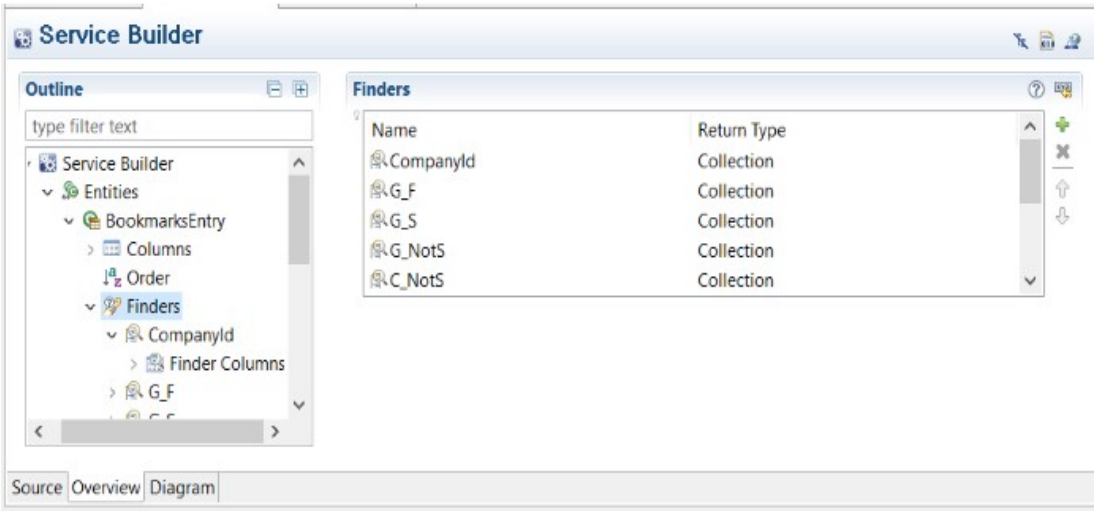

Рис. 4. Створення об’єктів Finder за допомогою Liferay Developer Studio

Після запуску Service Builder, він генерує пов’язані з пошуком методи (наприклад, fetchByGroupId, findByGroupId, removeByGroupId, countByGroupId) для організацій у класах *Persistence та *PersistentImpl. Перший із цих класів - інтерфейс, другий - це його реалізація.

Наприклад, програма Bookmarks генерує свої методи пошуку сутностей у класах -Persistence, що знаходяться в /bookmarks-api/src/main/java/com/liferay/ bookmarks/ service/persistent папці та класах -PersistenceImpl в /bookmarks-service/src/main/java/com/liferay/bookmarks/service/ persistent/impl папці. Щоб створити сервіс з файлу service.xml, використовується Liferay Developer Studio або вікно терміналу.

Використання Liferay Developer Studio. Відкриваючи програму Package Explorer, необхідно відкрити файл service.xml з кореневої папки*- сервісного модуля. За замовчуванням файл відкривається в Service Builder Editor. При цьому потрібно перебувати в режимі огляду. Потім натиснути кнопку “Будувати послуги” у верхньому правому куті зображення. Приклад панелі зображення можна побачити на рис. 5.

Ще один простий спосіб запустити програму Service Builder - це натиснути правою кнопкою миші на ім’я проекту в Package Explorer, а потім вибрати Liferay → build-service.

Використання терміналу. У вікні терміналу потрібно перейти до кореневої папки проекту модуля, яка повинна знаходитись у каталозі модулів Liferay Workspace. Liferay Workspace пропонує середовище побудови Gradle або Maven. Для проектів Gradle потрібно ввести таку команду в кореневу папку проекту модуля, щоб створити сервіси:

gradlew buildService

Для Maven відповідно:

mvn service-builder:build.

Коли послуга успішно створена, у вікні терміналу з’являється повідомлення BUILD SUCCESSFUL. Також можна побачити, що у проекті було створено велику кількість файлів. Ці файли включають рівень моделі, рівень обслуговування та рівень стійкості.

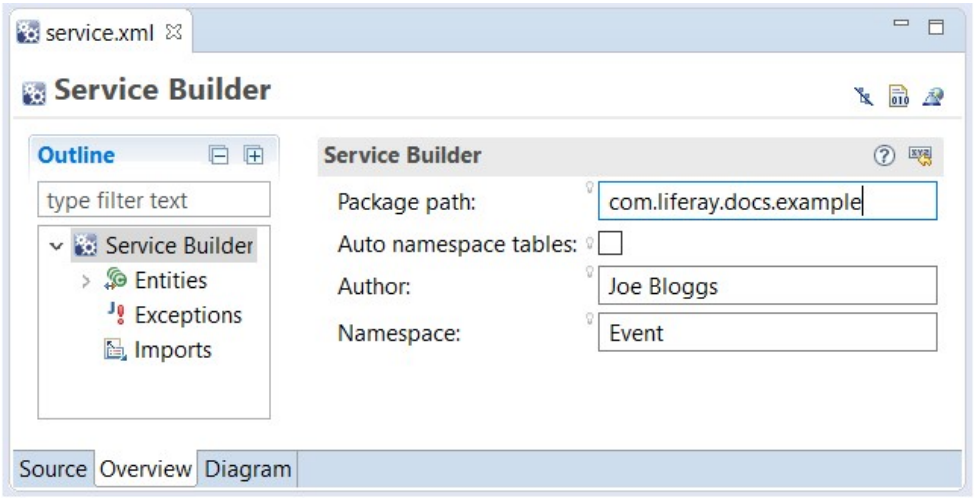

Рис.5. Режим огляду в редакторі

Кожен файл, який генерує Service Builder, збирається з пов'язаного шаблону FreeMarker. Ви можете знайти шаблони FreeMarker Service Builder у модулі portal-tools-service-builder. Наприклад, якщо ви хочете дізнатися, як створюється файл *ServiceImpl.java, просто подивіться на шаблон service_impl.ftl.

З усіх класів, згенерованих Service Builder, лише три слід змінювати вручну: *LocalServiceImpl, *ServiceImpl та *Impl. Якщо вручну модифікувати інші класи, зміни будуть перезаписані наступного разу, коли Service Builder буде запущено. Щоразу, коли розробники додають методи, видаляють методи або змінюють підпис методу класу *LocalServiceImpl, класу *ServiceImpl або *Impl, слід запустити Service Builder знову для відновлення порушених інтерфейсів та служби JAR.

Висновки і перспективи подальших досліджень

Проаналізувавши компоненти для розробки сайтів MVC Portlet [9, 10] було проаналізовано найбільш вдалу конфігурацію WEB-модулю з уточненням особливостей OSGi мета-даних. При аналізуванні логіки контролера MVC портлета отримані приклади (для випадків з малою кількістю функціоналу у додатка) як реалізувати всі основні методи для опрацювання дій, візуалізації та опрацювання ресурсів у одному портлеті. А також надана конфігурація шару представлень (JSP) для виклику методів портлета. Для додатків з великою кількістю функціональності реалізуються окремі команди, що використовуються в залежності від задачі, яку необхідно виконати на стороні сервера.

Функціонал для MVC команди дії містить пояснення параметрів, компонентів та властивостей, що необхідні для застосування команди дії, а також пояснення як прив'язати URL посилання, що створиться в JSP, до конкретного класу команди дії.

Функціонал для MVC команди візуалізації містить приклад впровадження MVCRenderCommand, створення URL до портлету візуалізації і, відповідно, реєстрації MVCRenderCommand для ідентифікації під час виклику з URL посилання. А також приклади оперування різними кінцевими візуалізаціями макетів (JSP) в залежності від описаної логіки команди.

Функціонал для MVC команди ресурсів містить принципи реєстрації команди ресурсів в OSGi модулі для ідентифікації сервісу під час ініціювання запиту на обробку ресурсів зі сторони сервера. А також пояснення реєстрації ідентифікатора для використання його в JSP і коротке пояснення коректного іменування класу команди.

Функціонал для Service Builder компоненту містить опис для коректного створення service.xml файлу, що використовується для опису повного опису сутностей, які після запуску компонент автоматично створить у підключеній базі даних, а також опис коректного глобального налаштування Service Builder в проекті. Реалізація містить визначення сутностей сервісу і, відповідно, стовпців сутностей, а також визначення відносин між сутностями і тип сортування сутностей під час отримання їх екземплярів з бази даних. В кінці зазначено визначення методів пошуку екземплярів сутностей і засобів для створення сервісу в результаті всіх попередніх налаштувань.

Таким чином проаналізовані функціонали можна використовувати для розробки Web-додатку, використовуючи основні компоненти порталу Liferay.

Література

1. Spring 4 для профессионалов: [навчальний посібник] / Кріс Шеффер, Кларенс Хо та Роб Харроп – 4-е видання – М.: "Вильямс", 2015. – 752 с. – ISBN 978-5-8459-19992-2. **2. Liferay**, Inc. Introduction to Liferay Development. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360018156791-Introduction-to-Liferay-Development. **3. Soft-Werke Co. Ltd**, What is Adobe Experience Manager. [Електронний ресурс] – Режим доступу: https://soft-werke.com/en/services/aem/ **4. Smile**, Що таке Jahia?

[Електронний ресурс] – Режим доступу: https://smile-ukraine.com/ua/jahia/introduction. **5. Wikipedia**, WebSphere. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/WebSphere. **6. Wikipedia**, SharePoint. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/SharePoint. **7. Using Drupal**: [навчальний посібник] / Анжела Байрон, Еддісон Беррі, – 1-е видання – O'REILLY, 2010. – 494 с. – ISBN 978-0-596-51580-5. **8. Liferay**, Inc. Service Builder. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/sections/360002519091-Service-Builder. **9. Liferay**, Inc. Liferay MVC Portlet. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360018159451-Liferay-MVC-Portlet. **10. Liferay**, Inc. Creating an MVC Portlet. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360017880432-Creating-an-MVC-Portlet-#creating-an-mvc-portlet.

11. Liferay, Inc. Spring MVC. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360017880512-Spring-MVC. **12. Liferay**, Inc. MVC Action Command. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360017880452-MVC-Action-Command.

13. Liferay, Inc. MVC Render Command. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360018159471-MVC-Render-Command.

14. Liferay, Inc. MVC Resource Command. [Електронний ресурс] – Режим доступу: https://help.liferay.com/hc/en-us/articles/360018159491-MVC-Resource-Command. **15. Robert Chen**. Liferay Beginner's Guide. [Текст] – Packt Publishing Ltd – 2011. – с. 325. **16. Samir Bhatt**. Liferay Portal Performance Best Practices. [Текст] – Packt Publishing Ltd – 2013. – с. 122. **17. Jonas X. Yuan**. Liferay Portal 6 Enterprise Intranets. [Текст] – Packt Publishing Ltd – 2010. – с. 74.

МЕТОДИКА РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ НА ОСНОВЕ ПОРТАЛА LIFERAY

Юрий Васильевич Кравченко *(доктор технических наук, профессор)*[1]
Максим Георгиевич Тищенко *(кандидат технических наук)*[1]
Александр Александрович Шапран[1]
Евгений Александрович Судников[1]
Валентин Геннадьевич Твардовский[2]

[1]***Национальный университет обороны Украины имени Ивана Черняховского, Киев, Украина***
[2]***Киевский национальный университет имени Тараса Шевченко, Киев, Украина***

Сервисы, доступные через веб-интерфейс, стали неотъемлемой частью современного мира. В связи с ростом числа Web-приложений, их поддержка и администрирование становится все более сложной задачей. Таким образом, рано или поздно практически перед любой организацией стоит задача интеграции этих сервисов.

Оптимальным вариантом решения описанной выше проблемы является развертывание корпоративного портала, который обеспечит централизованный доступ ко всем веб-сервисам организации.

В подавляющем большинстве разработчики Web-приложений для решения указанных задач используют корпоративный портал Liferay, который является мощной и современной платформой с открытым исходным кодом. В настоящее время является одним из наиболее популярных решений при построении корпоративных порталов.

Но у разработчиков возникает проблема с наличием в свободном доступе материалов и документации, которая в полной мере раскрывала весь возможный спектр реализации функционала платформы.

Следовательно, существует необходимость в детальном анализе функционала и примеров реализации на основе портала Liferay для разработки Web-приложений.

В статье проведен анализ базового принципа разработки Web-приложений, его реализацию в исследуемом корпоративном портале Liferay, а также в доступных для разработчиков модулях. Раскрыто функционал инструмента для разработки проектов Liferay Digital Experience Platform (DXP).

Проанализирована базовая структура основных компонентов портала для разработки корпоративных задач Liferay и на основе этой информации проанализированы эффективные примеры реализации этих компонентов, которые можно применять для создания структуры Web-приложений любого типажа и с любыми задачами.

Ключевые слова: *Liferay; Web-приложение; Model View Controller; Liferay Digital Experience Platform.*

METHOD OF DEVELOPING WEB APPLICATIONS BASED ON LIFERAY PORTAL

Yurii Kravchenko *(Doctor of Technical Sciences, Professor)*[1]
Maksym Tyshchenko *(Candidate of Technical Sciences)*[1]
Oleksandr Shapran[1]
Yevhen Sudnikov[1]
Valentyn Tvardovskyi[2]

[1]***National Defence University of Ukraine named after Ivan Cherniakhovsky, Kyiv, Ukraine***
[2]***Taras Shevchenko National University of Kyiv, Kyiv, Ukraine***

Services accessible through a web interface have become an integral part of the modern world. With the growing number of Web applications, their maintenance and administration are becoming an increasingly difficult task. Thus, sooner or later almost any organization faces the task of integrating these services.

The best solution to the problem described above is to deploy a corporate portal that will provide centralized access to all web services in the organization.

The vast majority of Web application developers use the enterprise portal Liferay, which is a powerful and modern open-source platform, to solve these problems. Currently, it is one of the most popular solutions for building corporate portals.

However, developers have a problem with the availability of materials and documentation that would fully reveal the full range of possible implementation of the functionality of the platform.

Therefore, there is a need for a detailed analysis of the functionality and implementation examples based on the Liferay portal for the development of Web applications.

The article analyzes the basic principle of Web application development, its implementation in the researched corporate portal Liferay, as well as in the modules available to developers. The functionality of the Liferay Digital Experience Platform (DXP) project development tool is revealed.

The basic structure of the main components of the portal for the development of corporate tasks Liferay is analyzed and based on this information, effective examples of implementation of these components are analyzed, which can be used to create a structure of Web-applications of any type and with any tasks.

Keywords: *Liferay; Web Application; Model View Controller; Liferay Digital Experience Platform.*

References

1. Kris Sheffer (2015), Spring 4 for professionals: [Spring 4 dlja professionalov], Moscow, Vil'jams, 752 p. **2. Liferay**, Inc. Introduction to Liferay Development. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360018156791-Introduction-to-Liferay-Development. **3. Soft-Werke Co. Ltd**, What is Adobe Experience Manager.[Електронний ресурс] – Режим доступу: https://soft-werke.com/en/services/aem/ **4. Smile**, Що таке Jahia? [Electronic resource]. URL: https://smile-ukraine.com/ua/jahia/introduction. **5. Wikipedia**, WebSphere. [Electronic resource]. URL: https://ru.wikipedia.org/wiki/WebSphere. **6. Wikipedia**, SharePoint. [Electronic resource]. URL: https://uk.wikipedia.org/wiki/SharePoint. **7. Anzhela Bairon, Eddison Berri** (2010), Using Drupal: O'REILLY, 494 p. **8. Liferay**, Inc. Service Builder. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/sections/360002519091-Service-Builder. **9. Liferay**, Inc. Liferay MVC Portlet. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360018159451-Liferay-MVC-Portlet. **10. Liferay**, Inc. Creating an MVC Portlet. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360017880432-Creating-an-MVC-Portlet-#creating-an-mvc-portlet.

11. Liferay, Inc. Spring MVC. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360017880512-Spring-MVC. **12. Liferay**, Inc. MVC Action Command. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360017880452-MVC-Action-Command.

13. Liferay, Inc. MVC Render Command. [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360018159471-MVC-Render-Command.

14. Liferay, Inc. MVC Resource Command [Electronic resource]. URL: https://help.liferay.com/hc/en-us/articles/360018159491-MVC-Resource-Command.

15. Robert Chen, (2011). Liferay Beginner's Guide. Packt Publishing Ltd, p. 325. **16. Samir Bhatt** (2013), Liferay Portal Performance Best Practices. Packt Publishing Ltd, p. 122. **17. Jonas X. Yuan**, (2010), Liferay Portal 6 Enterprise Intranets, Packt Publishing Ltd, p. 74.