Лабораторна робота №4

# Створення плагінів Плагін Service Provider

Кішкін Олександр.

8.1212

1. **Проаналізувати** [**статтю**](https://javascript.plainenglish.io/building-vue-enterprise-application-part-2-services-f7ec400190e7)**.**

У статті розглядається розробка фронтенд-додатку на Vue.js з використанням принципів чистої архітектури. Стаття розповідає про створення сервісів, які відповідають за виконання основних бізнес-операцій. Основні пункти статті включають:

1. Розгляд поняття сервісу в контексті Vue.js та чистої архітектури, де сервіс описується як щось, що виконує основні бізнес-операції, і наголошується на тому, що сервіси не повинні мати залежності від деталей, таких як фреймворки або UI.
2. Початковий аналіз вимог для функціональності блогу, включаючи можливість перегляду статей та залишення коментарів.
3. Запуск розробки з використанням підходу розробки через тести, де автор створює тести для функціональності сервісу перед написанням коду для забезпечення правильної роботи.
4. Розгляд структури папок і файлів для організації сервісу та його тестів.
5. Обговорення важливості ін'єкції даних у сервіс та рекомендація внедрити дані замість жорсткої залежності від файлів чи інших джерел.
6. Розгляд паттерну сервісного провайдера, який допомагає інкапсулювати ініціалізацію сервісів та надавати їх іншим частинам додатку.

Стаття містить добре структурований код та приклади, які допомагають зрозуміти концепції та практичні аспекти розробки Vue.js додатків з використанням чистої архітектури. Ця стаття є частиною серії і корисна для розробників, які цікавляться створенням великих і складних додатків на Vue.js.

1. **Створити провайдер для сервісів створених у попередній лабораторній роботі.**

Для створення провайдера – створимо файл TransliterationProvider.js

В файлі напишемо функцію транслітерації.

// TransliterationProvider.js

export default {

  transliterate(text) {

    const transliterationMap = {

      'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e', 'ё': 'yo',

      'ж': 'zh', 'з': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm',

      'н': 'n', 'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't', 'у': 'u',

      'ф': 'f', 'х': 'kh', 'ц': 'ts', 'ч': 'ch', 'ш': 'sh', 'щ': 'sch', 'ъ': '',

      'і': 'y', 'ь': '', 'э': 'e', 'ю': 'yu', 'я': 'ya'

    };

    return text.split('').map(char => transliterationMap[char] || char).join('');

  }

}

Вносимо зміни в файл TasksList.vue.

<script setup>

import TransliterationProvider from './TransliterationProvider';

defineProps({

  tasks: Array,

  tab: String,

  editedTask: Object

});

defineEmits(['removeTask', 'updateTask', 'submit', 'changeStatus'])

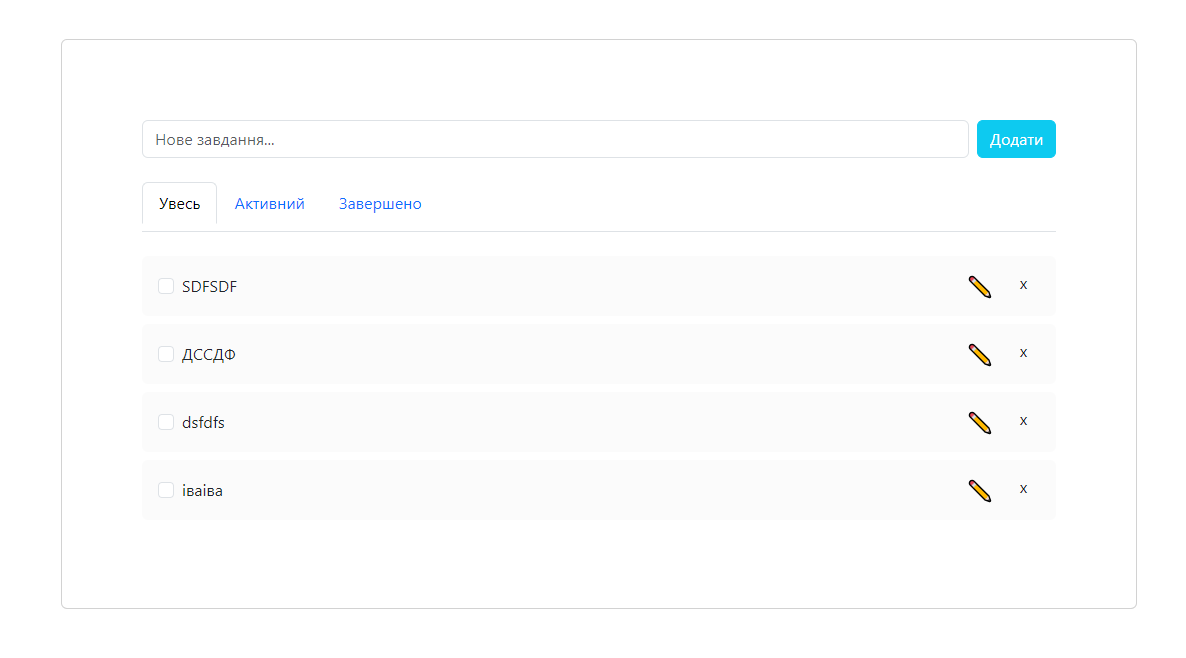
const transliterateTitle = (title) => {

  return TransliterationProvider.transliterate(title);

};

</script>

Перевіряємо роботу.



1. **Зберігти зміни в новому репозиторію з назвою mjsf-mag-lab4.**

https://github.com/KishkinA/mjsf-mag-lab4

1. **Надати посилання на репозиторій у якості звіту.**

### Контрольні питання

1. **Що таке плагіни в контексті Vue.js 3 і як вони розширюють функціональність Vue?**

Плагіни в контексті Vue.js 3 - це спеціальні бібліотеки або розширення, які дозволяють розширити функціональність Vue.js. Вони надають можливість додавати нові функції, компоненти, директиви, методи та інші ресурси до вашого додатку Vue без необхідності вносити зміни в основний фреймворк Vue.js.

Ось деякі способи, якими плагіни можуть розширювати функціональність Vue.js:

1. **Додавання глобальних компонентів:** Плагіни можуть додавати глобальні компоненти, які будуть доступні у всьому додатку Vue. Наприклад, плагін може додати компонент для сповіщень або модальних вікон.
2. **Створення кастомних директив:** Ви можете використовувати плагіни для створення кастомних директив, які можуть бути використані у ваших Vue компонентах.
3. **Розширення функціональності Vue іншими бібліотеками:** Плагіни дозволяють інтегрувати Vue.js з іншими бібліотеками, такими як Vuex для керування станом, Vue Router для маршрутизації або Axios для роботи з HTTP-запитами.
4. **Додавання методів до прототипу Vue:** Плагіни можуть розширити прототип Vue і додати методи, які будуть доступні у всіх екземплярах Vue у вашому додатку.
5. **Надання налаштувань та параметрів:** Плагіни можуть надавати налаштування, які дозволяють користувачам настроювати їхню роботу, наприклад, змінювати поведінку чи стилізацію компонентів.

Для встановлення та використання плагіна в Vue.js 3 ви зазвичай встановлюєте плагін за допомогою npm або yarn, імпортуєте його в свій код та використовуєте за допомогою методів та функцій, які надає плагін.

Плагіни роблять Vue.js більш гнучким та розширюють його можливості, дозволяючи створювати багато різних розширень та додатків для вашого додатку Vue.

1. **Як встановити плагін?**

Для встановлення плагіна Vue.js вам зазвичай потрібно виконати наступні кроки:

1. **Створіть новий проект Vue.js:** Якщо у вас ще немає проекту Vue.js, спочатку створіть новий проект, використовуючи Vue CLI або будь-який інший спосіб, який ви виберете. Наприклад, ви можете створити проект Vue CLI командою vue create my-project, де my-project - це назва вашого проекту.
2. **Перейдіть в папку проекту:** Відкрийте командний рядок або термінал, перейдіть до папки вашого проекту, використовуючи команду cd my-project, де my-project - це назва вашого проекту.
3. **Встановіть плагін:** Ви можете встановити плагін Vue.js за допомогою пакетного менеджера npm або yarn, використовуючи команду npm install plugin-name або yarn add plugin-name, де plugin-name - це ім'я плагіна, який ви хочете встановити. Наприклад, якщо ви хочете встановити плагін vue-router, ви можете використовувати команду npm install vue-router або yarn add vue-router.
4. **Імпортуйте і використовуйте плагін в вашому проекті:** Після встановлення плагіну ви повинні імпортувати його в вашому коді. Наприклад, для vue-router ви можете додати імпорт до вашого файлу Vue-компонента:

import VueRouter from 'vue-router';

Після імпортування ви можете використовувати плагін у вашому проекті, налаштовувати його і використовувати його функціональність.

1. **Конфігурація та використання:** Зазвичай плагіни Vue.js мають документацію, яка надає вам інформацію про те, як налаштувати і використовувати їх у вашому проекті. Слідкуйте інструкціями з документації плагіна, щоб коректно налаштувати і використовувати його функціональність.

Ці кроки допоможуть вам встановити плагін Vue.js у вашому проекті і почати використовувати його функціональність в залежності від вашого використання.

1. **Які два способи визначення плагіна підтримуються в Vue.js 3?**

Vue.js 3 підтримує два способи визначення плагінів:

1. **Глобальні плагіни (Global Plugins):** Цей спосіб включає встановлення плагіна глобально для всього додатку Vue. Щоб встановити глобальний плагін, ви використовуєте глобальний метод app.use() у вашому основному файлі додатку (зазвичай, main.js). Наприклад, якщо ви хочете встановити плагін Vue Router:

import { createApp } from 'vue';

import { createRouter, createWebHistory } from 'vue-router';

import App from './App.vue';

const router = createRouter({

history: createWebHistory(),

routes: [

// ваші маршрути

],

});

const app = createApp(App);

app.use(router); // Встановлюємо плагін Vue Router глобально

app.mount('#app');

Плагін, встановлений таким чином, буде доступний у всьому додатку Vue.

1. **Локальні плагіни (Local Plugins):** Локальні плагіни встановлюються на рівні окремого компонента. Це означає, що ви можете встановити плагін лише для певного компонента та його дочірніх компонентів. Щоб встановити локальний плагін, ви використовуєте метод use() всередині компонента. Наприклад:

import { ref, onMounted } from 'vue';

const MyComponent = {

setup() {

const myPluginInstance = useMyPlugin(); // Встановлюємо локальний плагін

// Решта коду компонента

},

};

Локальний плагін буде доступний лише для цього компонента та його дочірніх компонентів, і це дозволяє створювати ізольовані розширення для конкретних компонентів.

Обираючи між глобальними та локальними плагінами, ви можете керувати областю видимості та доступністю вашого плагіна в додатку Vue, вибираючи найбільш підходящий під ваші потреби спосіб визначення плагіна.

1. **Які можливі сценарії використання плагінів Vue?**

Плагіни Vue.js можуть бути використані для різних сценаріїв і розширювати функціональність вашого додатку Vue. Ось декілька можливих сценаріїв використання плагінів:

1. **Маршрутизація (Vue Router):** Ви можете використовувати плагін Vue Router для додавання маршрутизації до вашого додатку Vue, щоб створити односторінковий додаток (SPA) з різними сторінками та маршрутами.
2. **Управління станом (Vuex):** Плагін Vuex дозволяє вам керувати станом додатку і виконувати декларативне управління станом між компонентами, спрощуючи управління даними.
3. **Робота з HTTP-запитами (axios, vue-resource):** Ви можете встановити плагіни для роботи з HTTP-запитами, такі як Axios або Vue Resource, для виконання запитів до сервера і обробки даних.
4. **Валідація даних (VeeValidate):** Плагіни для валідації даних, наприклад, VeeValidate, дозволяють вам валідувати дані, які вводять користувачі, і відображати повідомлення про помилки.
5. **Розширення компонентів:** Ви можете використовувати плагіни для додавання нових функцій до ваших Vue компонентів, таких як глобальні компоненти, кастомні директиви, міксіни тощо.
6. **Сповіщення та модальні вікна (Vue-Toastification, vue-modal-dialog):** Плагіни можуть додавати компоненти для відображення сповіщень, попереджень та модальних вікон.
7. **Міжнародизація (vue-i18n):** Плагін vue-i18n дозволяє локалізувати ваш додаток, додаючи підтримку різних мов та перекладів.
8. **Аналітика (vue-analytics, vue-metrics):** Плагіни для аналітики допомагають вам відстежувати та аналізувати взаємодію користувачів з вашим додатком.
9. **Завдання рендерингу на стороні сервера (Nuxt.js):** Nuxt.js - це плагін для Vue.js, який додає можливість рендерити сторінки на стороні сервера, що підвищує SEO та швидкодію додатку.
10. **Інтеграція з іншими бібліотеками та фреймворками:** Плагіни можуть бути використані для інтеграції з іншими бібліотеками, такими як Three.js для роботи з 3D-графікою або D3.js для візуалізації даних.

Це лише кілька прикладів, існує безліч інших плагінів, які можна використовувати для розширення функціональності Vue.js в залежності від конкретних потреб вашого проекту. Використання плагінів дозволяє значно полегшити розробку та розширити можливості вашого додатку.

1. **Як передати параметри конфігурації в плагін під час встановлення?**

При встановленні плагіна в Vue.js ви можете передати параметри конфігурації, які плагін може використовувати для налаштування своєї роботи. Це допомагає зробити плагін більш гнучким і адаптованим до потреб вашого додатку. Для передачі параметрів конфігурації ви можете використовувати об'єкт конфігурації або інші значення в залежності від того, яким чином плагін реалізує обробку конфігурації.

Ось приклад того, як передати параметри конфігурації при встановленні плагіна:

1. **За допомогою об'єкта конфігурації:**

Передбачаємо, що плагін приймає об'єкт конфігурації під час встановлення.

// main.js

import { createApp } from 'vue';

import MyPlugin from 'my-plugin';

const app = createApp(App);

const pluginConfig = {

option1: 'value1',

option2: 'value2',

};

app.use(MyPlugin, pluginConfig); // Передаємо об'єкт конфігурації при встановленні

У плагіні можна отримати цей об'єкт конфігурації та використовувати його параметри:

// my-plugin.js

export default {

install(app, options) {

const { option1, option2 } = options;

// Використовуємо параметри конфігурації

},

};

 **За допомогою окремих параметрів:**

В інших випадках плагін може приймати окремі параметри замість об'єкта конфігурації.

// main.js

import { createApp } from 'vue';

import MyPlugin from 'my-plugin';

const app = createApp(App);

const option1 = 'value1';

const option2 = 'value2';

app.use(MyPlugin, option1, option2); // Передаємо окремі параметри конфігурації

У плагіні можна отримати ці параметри та використовувати їх:

// my-plugin.js

export default {

install(app, option1, option2) {

// Використовуємо параметри конфігурації

},

};

Важливо враховувати, що інтерфейс передачі параметрів конфігурації може варіюватися в залежності від того, яким чином плагін був створений та як він очікує отримувати параметри. Рекомендується докладно ознайомитися з документацією плагіна або бібліотеки, яку ви використовуєте, щоб правильно передавати параметри конфігурації.

1. **Як можна використовувати inject для надання функції або атрибута користувачам плагіна?**

Метод inject в Vue 3 використовується для надання функцій або атрибутів користувачам плагіна або компоненту. Цей підхід дозволяє вам створювати більшу гнучкість та можливість керування взаємодією між плагінами та компонентами в додатку Vue.

Ось кілька кроків, які потрібно виконати для використання inject для надання функцій або атрибутів користувачам плагіна:

1. **Створіть плагін або компонент, який буде надавати функції або атрибути.**

Приклад створення простого плагіну, який надає функцію:

// my-plugin.js

export default {

install(app) {

app.provide('myFunction', () => {

// Ваша функція

});

},

};

 **Встановіть плагін в своєму додатку Vue.**

Після створення плагіну встановіть його в своєму додатку:

import { createApp } from 'vue';

import MyPlugin from './my-plugin';

import App from './App.vue';

const app = createApp(App);

app.use(MyPlugin); // Встановлюємо плагін

app.mount('#app');

 **Використовуйте inject для отримання функції або атрибута в вашому компоненті.**

У вашому компоненті ви можете використовувати inject, щоб отримати надану функцію або атрибут. Наприклад:

// MyComponent.vue

export default {

setup() {

const myFunction = inject('myFunction');

// Використовуємо функцію

myFunction();

// Решта коду компонента

},

};

1. Зауважте, що параметр, переданий в inject (у цьому прикладі, 'myFunction'), повинен відповідати тому, яким чином ви використовуєте provide в вашому плагіні. Ім'я параметра в inject відповідає ключу, який ви вказали при виклику provide.

Цей підхід дозволяє зручно обмінюватися функціями, атрибутами та іншими ресурсами між компонентами та плагінами в додатку Vue, і він може бути корисним для створення розширень та додатків, які надають спеціальну функціональність.