Livewire vs Inertia.js

關於如何選擇**全端框架**這檔事

{LARAVEL x VUE} Conf Taiwan 2021

Lucas Yang

喜歡探索有趣的事物。嗜好是看動漫。

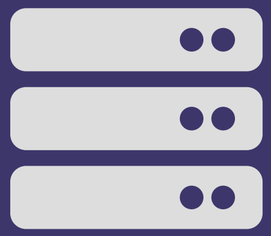
目前就讀國立空中大學 - 管理與資訊學系

使用框架：Laravel / Vue.js / Tailwind CSS

ycs77

lucas-yang.vercel.app

現代 Web 應用架構



伺服器端渲染

(後端渲染)

用戶端渲染

(前端渲染)

經典後端網站架構

路由(後端路由)

+

資料(ORM)

+

渲染(後端模板)

=

後端應用

(Laravel, Rails)

「全端網站」的好朋友

SPA 時代來臨，網站前後端分離

路由(前端路由)

+

資料(API)

+

渲染(前端組件)

=

前端應用

(Vue.js, React, Angular)

以 **Laravel** & **Vue** 舉例，「全端」需包辦以下

- 開發 API (REST API、GraphQL)
- 管理 2 個 Git 專案 (SPA & API)
- 前端路由 (vue-router)
- 身分驗證 (JWT…)
- 狀態管理
- CORS

全端：建立**現代全端網站**很複雜

Laravel 的全端解決方案

Livewire

Inertia.js

- **Laravel 的全端框架**
- 核心概念：組件
- 使用語言：PHP & Blade
- 資料輸入雙向綁定 (前後端使用 AJAX 溝通)
- SEO 友好

計數器

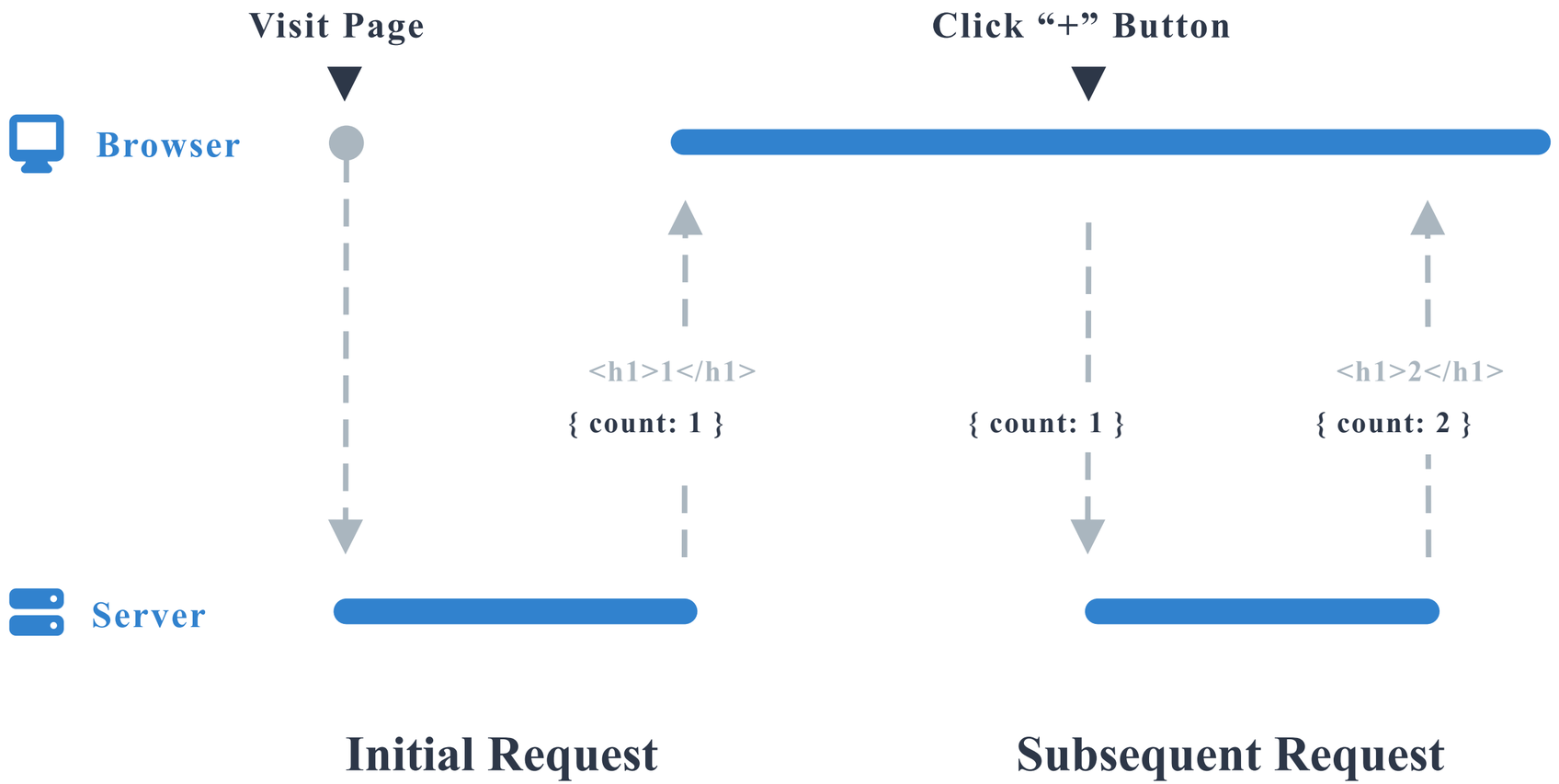
Livewire 組件

```php
// app/Http/Livewire/Counter.php
use Livewire\Component;

class Counter extends Component
{
    public $count = 0;

    public function increment()
    {
        $this->count++;
    }

    public function render()
    {
        return view('livewire.counter');
    }
}
```

Livewire 視圖

```blade
<!-- resources/views/livewire/counter.blade.php -->
<div style="text-align: center">
    <button wire:click="increment">+</button>

    <h1>{{ $count }}</h1>
</div>
```

運作原理


Visit Page
Click “+” Button
Browser
<h1>1</h1>
{ count: 1 }
{ count: 1 }
<h1>2</h1>
{ count: 2 }
Server
Initial Request
Subsequent Request

雙向綁定

Livewire 組件

```php
use Livewire\Component;

class DataBinding extends Component
{
    public $name = '';
    public $agree = false;
    public $languages = ['Laravel', 'Vue.js'];

    public function render()
    {
        return view('livewire.data-binding');
    }
}
```

Livewire 視圖

```blade
<div>
    <input type="text" wire:model="name">
    <input type="checkbox" wire:model="agree">我同意服務條款

    @if ($agree)
        <div>哈囉~ {{ strtoupper($name) }}</div>
    @else
        <div>{{ $name }} 請勾選同意服務條款</div>
    @endif

    <ul>
        @foreach ($languages as $language)
            <li>{{ $language }}</li>
        @endforeach
    </ul>
</div>
```

即時搜尋

Livewire 組件

```php
use App\Models\Language;
use Livewire\Component;

class Search extends Component
{
    public $search = '';

    protected $queryString = [
        'search' => ['except' => ''],
    ];

    public function render()
    {
        return view('livewire.search', [
            'languages' => Language::where('name', 'LIKE',
        ]);
    }
}
```

Livewire 視圖

```blade
<div>
    <input type="text" wire:model="search">

    <ul>
        @foreach ($languages as $language)
            <li>{{ $language->name }}</li>
        @endforeach
    </ul>
</div>
```

用 **PHP & Blade** 寫出 **JavaScript** 的效果

Blade vs Vue?

- 同時用 **Blade** 和 **Vue** 來組織視圖

```
<div class="container">
    <create-item :data="@json($data)" />

    <ul>
        @foreach ($items as $item)
            <li>{{ $item->content }}</li>
        @endforeach
    </ul>
</div>
```

- 視圖全權交給 **Vue** (View)

```
<items-list :data="@json($page)" />
```

路由(後端路由)

+

資料(ORM)

=

前端渲染

的

後端應用

渲染(前端組件)

+

inertia»

- Inertia 是開發 **經典後端應用** (Laravel, Rails) + **前端視圖** (Vue.js, React) 的方法
- 後端框架可以使用除了視圖的所有功能，如 Router、Controller、Middleware、身分驗證、授權、Eloquent ORM……等。
- 沒有 API、沒有前端路由、也沒有 SPA 應用的複雜性
- Inertia 定義了一組**協議** (規則)
- 提供多個實作 Inertia 的 前端框架套件 和 後端框架套件

HTTP 請求

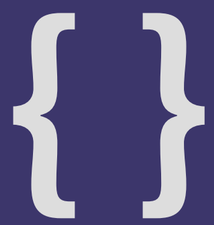
GET /events/80

HTTP 請求

HTML 響應

GET /api/events/80

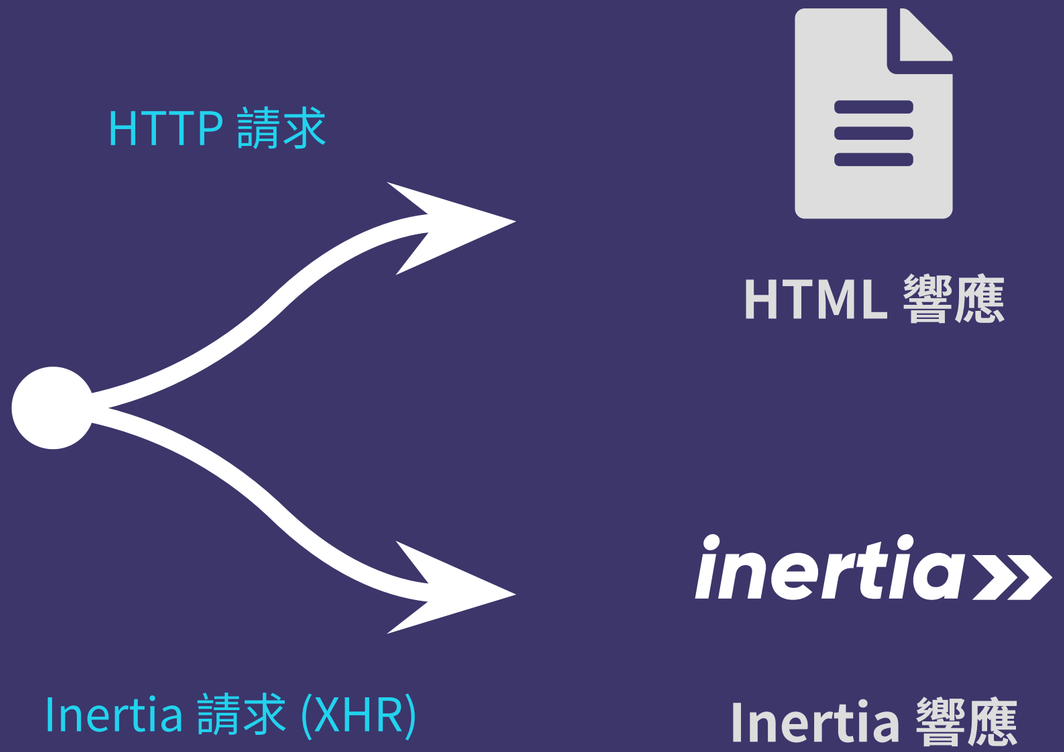
XHR 請求



JSON 響應

HTTP 請求 (Inertia)

GET **/events/80**



瀏覽頁面

- 回傳 HTML 頁面
- 基本的 CSS 和 JS
- 用來啟動 SPA 應用的 ** 根元素**
  (`data-page` 屬性中包含 **Inertia Page 物件**)

REQUEST

```
GET: http://example.com/events/80
Accept: text/html, application/xhtml+xml
```

RESPONSE

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
<head>
    <title>My app</title>
    <link href="/css/app.css" rel="stylesheet">
    <script src="/js/app.js" defer></script>
</head>
<body>

<div id="app" data-page='{"component":"Event","props":{"event":
{"id":80,"title":"Birthday party","start_date":"2019-06-
02","description":"Come out and celebrate Jonathan&amp;apos;s 36th birthday
party!"}},"url":"/events/80","version":"c32b8e4965f418ad16eaebba1d4e960f"}'>
</div>

</body>
</html>
```

Inertia 加載新頁面

- 點擊 <inertia-link> 組件，
- Inertia 會攔截點擊事件並發送一個 **XHR 請求**
  並回傳 **Inertia Page 物件**
- 然後更新當前頁面的 **前端組件** 和 **瀏覽紀錄**

Inertia Page 物件

- **component**：前端組件名稱
- **props**：前端組件 props (資料)
- **url**：頁面 URL
- **version**：當前資產版本
  (在 Laravel 中會自動使用 `mix-manifest.json` 生成的 Hash 值作為資產版本)

REQUEST

```
GET: http://example.com/events/80
Accept: text/html, application/xhtml+xml
X-Requested-With: XMLHttpRequest
X-Inertia: true
X-Inertia-Version: 6b16b94d7c51cbe5b1fa42aac98241d5
```

RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "component": "Event",
  "props": {
    "event": {
      "id": 80,
      "title": "Birthday party",
      "start_date": "2019-06-02",
      "description": "Come out and celebrate Jonathan's 36th birthday
party!"
    }
  },
  "url": "/events/80",
  "version": "c32b8e4965f418ad16eaebba1d4e960f"
}
```

發送 Inertia 請求

```
import { Inertia } from '@inertiajs/inertia'

Inertia.visit(url, {
  method: 'get',
  data: {},
  replace: false,
  preserveState: false,
  preserveScroll: false,
  only: [],
  headers: {},
  errorBag: null,
  forceFormData: false,
  onCancelToken: cancelToken => {},
  onCancel: () => {},
  onBefore: visit => {},
  onStart: visit => {},
  onProgress: progress => {},
  onSuccess: page => {},
  onError: errors => {},
  onFinish: visit => {},
})
```

```
import { Inertia } from '@inertiajs/inertia'

Inertia.get(url, data, options)
Inertia.post(url, data, options)
Inertia.put(url, data, options)
Inertia.patch(url, data, options)
Inertia.delete(url, options)
```

Inertia Link

```
import { Link } from '@inertiajs/inertia-vue3'

<Link href="/">首頁</Link>

// Method：POST / PUT / PATCH / DELETE
<Link href="/logout" method="post" as="button" type="button">登出</Link>
// 連結會渲染成
<button type="button">登出</button>

// post 資料
<Link href="/endpoint" method="post" :data="{ foo: bar }">Save</Link>

// 替換歷史紀錄
<Link href="/" replace>首頁</Link>

// 保持滾動條位置
<Link href="/" preserve-scroll>首頁</Link>

// Partial reloads（局部重載）
<Link href="/users?active=true" :only="['likes_count']">喜歡</Link>
```

Inertia Adapters (支援的框架)

Inertia 官方後端套件

- Laravel
- Rails

Inertia 官方前端套件

- Vue.js 2/3
- React
- Svelte

社區開發套件

Hello Inertia

- 渲染頁面
- Prop 傳資料

路由

```php
// routes/web.php
use Inertia\Inertia;

Route::get('/', function () {
    return Inertia::render('Home');
});
```

```php
// routes/web.php
use Inertia\Inertia;

Route::get('about', function () {
    return Inertia::render('About', [
        'name' => 'Lucas',
    ]);
});
```

VUE 組件

```vue
<!-- resources/js/pages/Home.vue -->
<template>
  <h1>首頁</h1>
</template>
```

```vue
<!-- resources/js/pages/About.vue -->
<template>
  <h1>關於</h1>
  <div>你好~ 我是 {{ name }}</div>
</template>

<script>
export default {
  props: {
    name: String,
  },
}
</script>
```

留言板 - 留言列表

資料庫拿資料

```php
Route::get('comments', function () {
    return Inertia::render('Comments', [
        'comments' => Comment::all()
            ->transform(fn (Comment $comment) => [
                'id' => $comment->id,
                'name' => $comment->name,
                'content' => $comment->content,
                'created_at' => $comment->created_at->form
            ]),
    ]);
});
```

```vue
<template>
  <h1>留言</h1>
  ...
  <ul>
    <li v-for="comment in comments">
      <h2>姓名：{{ comment.name }}</h2>
      <div>{{ comment.created_at }}</div>
      <div>內容：{{ comment.content }}</div>
    </li>
  </ul>
</template>

<script>
export default {
  props: {
    comments: Array,
  },
  // ...
}
</script>
```

留言板 - 表單送出

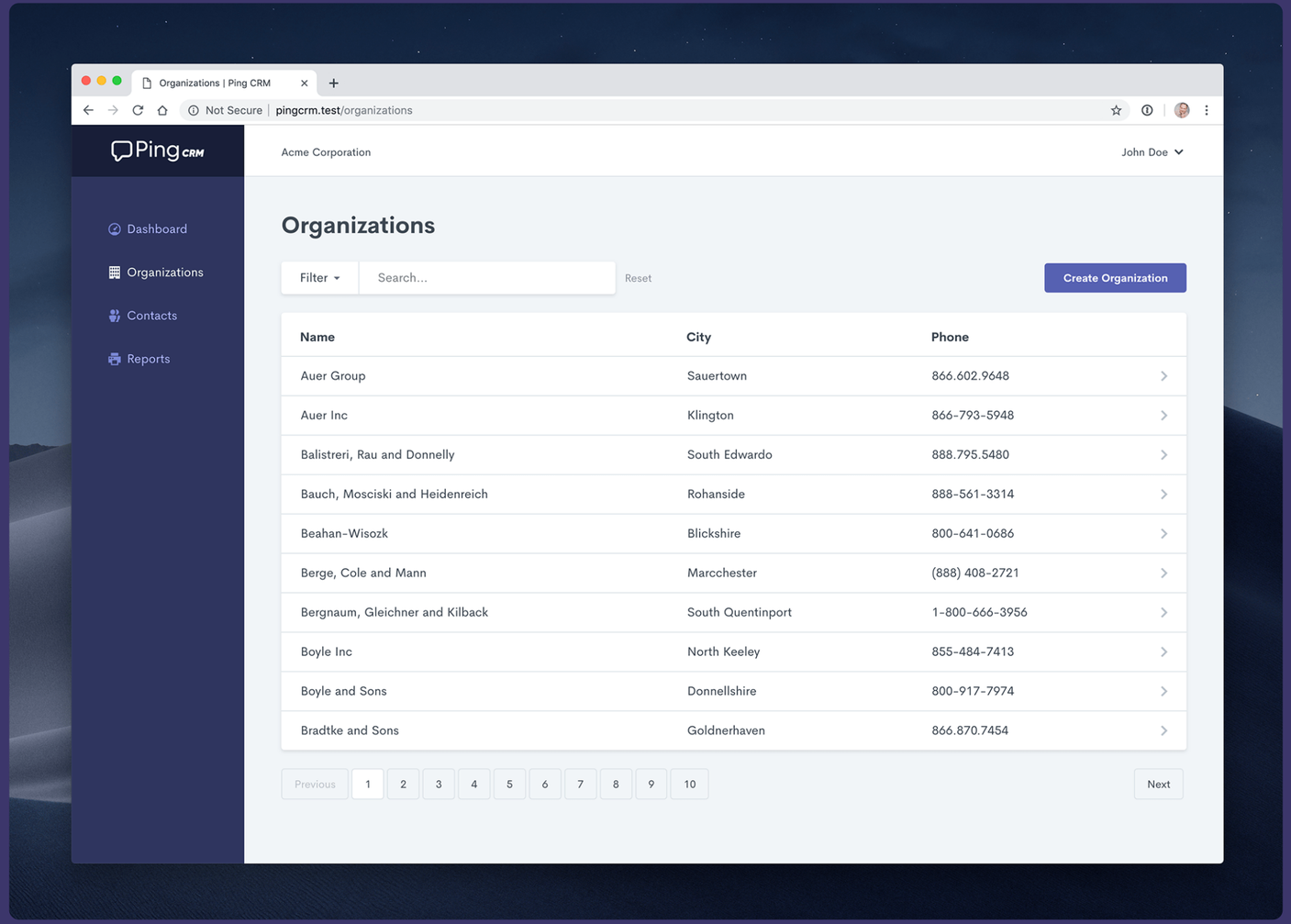
```php
Route::post('comments', function (Request $request) {
    $data = $request->validate([
        'name' => 'required',
        'content' => 'required',
    ]);

    Comment::create($data);

    return back();
});
```

留言板 - 表單送出

```
<template>
  <h1>留言</h1>

  <form @submit.prevent="storeComment">
    <div>姓名：<input type="text" v-model="form.name"></div
    <div v-if="form.errors.name" class="invalid">
      {{ form.errors.name }}
    </div>

    <div>留言：<textarea v-model="form.content"></textarea>
    <div v-if="form.errors.content" class="invalid">
      {{ form.errors.content }}
    </div>

    <button>送出</button>
  </form>
  ...
</template>
```

```
<script>
import { useForm } from '@inertiajs/inertia-vue3'

export default {
  // ...
  setup() {
    const form = useForm({
      name: '',
      content: '',
    })

    const storeComment = () => {
      form.post('/comments', {
        onSuccess: () => form.reset(),
      })
    }

    return { form, storeComment }
  },
}
</script>
```

Inertia Demo



Demo application | Ping CRM | GitHub

SSR (服務端渲染) - 優化 SEO 測試階段

1. 發送請求進入後端框架 - **Laravel**
2. Inertia 將 **Page 物件** 發送給本地 SSR Server，
   使用 Node.js 將當前頁面的 **前端組件** 渲染為 HTML - **Node.js**
3. 將 HTML 插入到響應中並回傳給用戶
4. SSR 模式下啟動前端框架 - **vue-server-renderer**
   (Vue、React、Svelte 都有處理 SSR 的工具)
5. SSR 渲染只會在進頁面時執行，之後就會以 SPA 模式運作

Ping CRM - SSR

TALL Stack

Tailwind CSS + **A**lpine.js + **L**aravel + **L**ivewire

VILT Stack

Vue.js + **I**nertia.js + **L**aravel + **T**ailwind CSS


Laravel Jetstream

Livewire vs Inertia.js

	Livewire	Inertia.js
框架	Laravel	支援 Inertia 的框架
視圖 (View)	Blade、PHP、Alpine.js	前端框架 Vue、React...
SEO	**易**	**難**
總結	用 PHP 寫出 JavaScript 的效果	前後端框架的膠水

Blade → **Livewire**

Vue.js → **Inertia.js**

參考資料

- Livewire 官網
- Inertia.js 官網
- Laravel Jetstream 官網
- TALL stack
- VILT Stack
- 前端渲染的後端應用 - 用 Inertia.js 拉近 Laravel 和 Vue.js 的距離