Programación de Servicios y Procesos

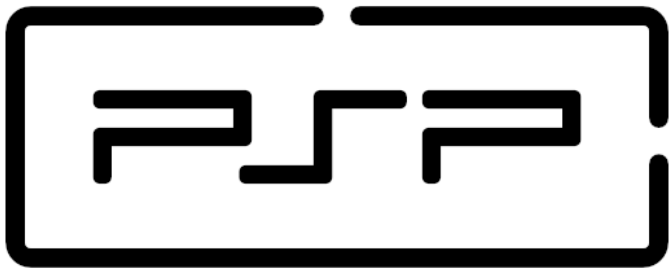
Apuntes de clase

Acceder al curso →

Introducción Tema 1

VuePress is composed of two parts: a **minimalistic static site generator** with a Vue-powered **theming system** and **Plugin API** , and a **default theme** optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.



Programación de Servicios y Procesos

Tema 3 - Hilos

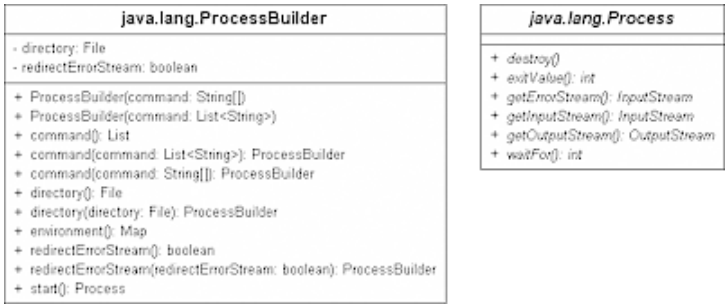
Threads y Runnable

I.E.S.
Doctor Balmis

9. Using Vue in Markdown by Vicente default theme

9.1. Browser API Access Restrictions new!

Because VuePress applications are server-rendered in Node.js when generAAating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



Pasaremos los datos necesarios para construir un objeto ISBN al constructor de Libro (en nuestro caso una cadena con el isbn) y será este quien lo cree. En el fondo esto es lo que sucede con un tipo valor como DateTime por su 'naturaleza'. De esta manera, además, el objeto Isbn13 que estoy creando desaparecerá junto con el libro.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component: ♡

```php
# Post Here Code Example
function base(a, b)
{
    print "hola";
}
```

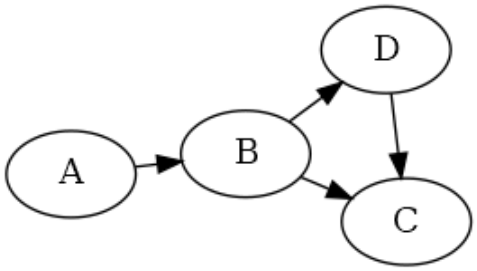
PHP Example **Java Title**

```php
# Post Here Code Example
```

```java
// Post Here Code Examples
```

```yaml
---
home: true
heroImage: /Logo_PSP.png
tagline: VuePress & Markdown
actionText: Access course →
actionLink: /unit1/
features:
- title: Feature 1 Título
  details: Feature 1 Description
- title: Feature 2 Title
  details: Feature 2 Description
- title: Feature 3 Title
  details: Feature 3 Description
footer: Made by Vicente Martínez with ♡
---

```



Hola
Cross-Relation

9.2. Browser API Access Restrictions

Because VuePress applications are server rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

9.2.1 Third level heading

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR- friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

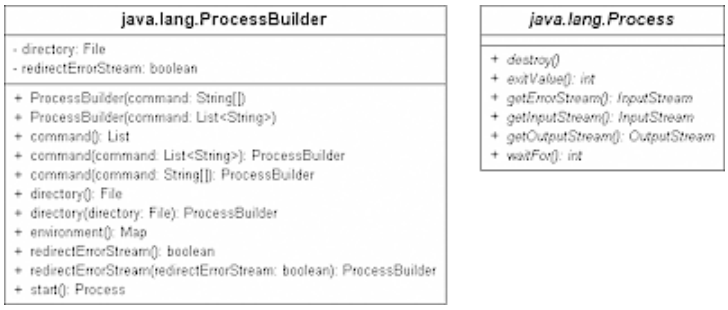
Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9. Using Vue in Markdown by Vicente default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

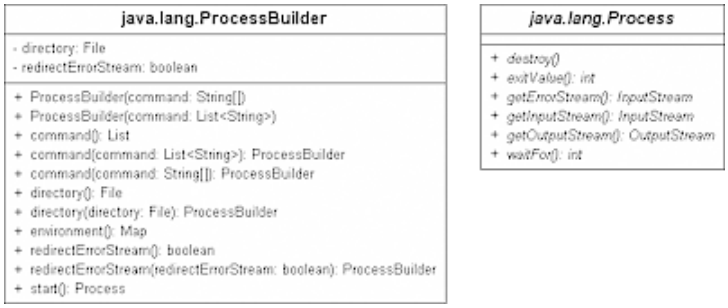
If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9. Using Vue in Markdown by Vicente v2

default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

Introducción Tema 2

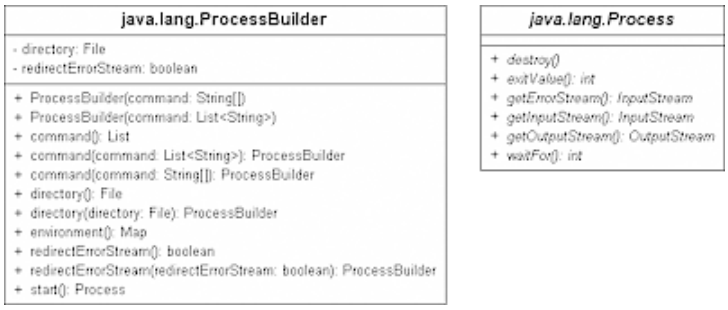
VuePress is composed of two parts: a **minimalistic static site generator** with a Vue-powered **theming system** and **Plugin API** , and a **default theme** optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

9. Using Vue in Markdown by Vicente default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

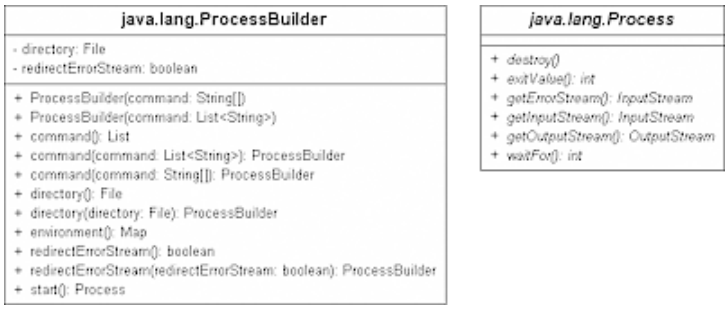
Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9. Using Vue in Markdown by Vicente default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

Process and Service Programming

Class notes

Access lessons →

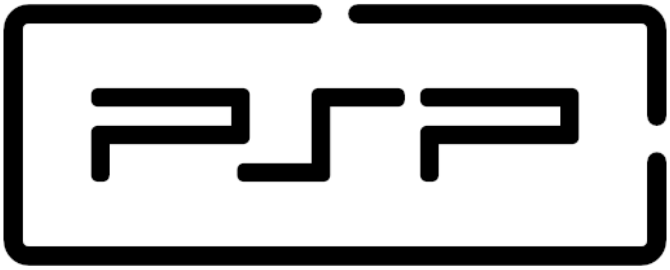
1. Introduction to concurrent programming

VuePress is composed of two parts: a **minimalistic static site generator** with a Vue-powered **theming system** and **Plugin API** , and a **default theme** optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

1.1 Another title

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.



Process and Service Programming

Unit 9.3 - Multiprogramming

Processes and threads

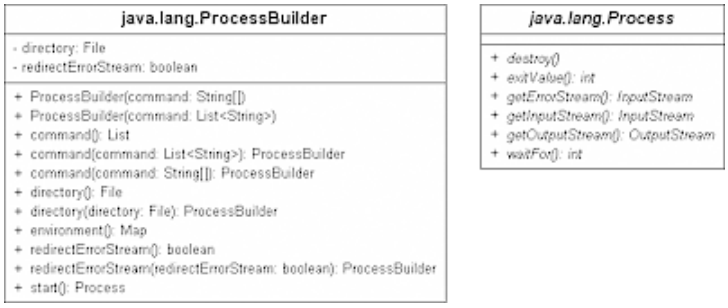
I.E.S.
Doctor Balmis

9. Using Vue in Markdown (print option)

default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generAAating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



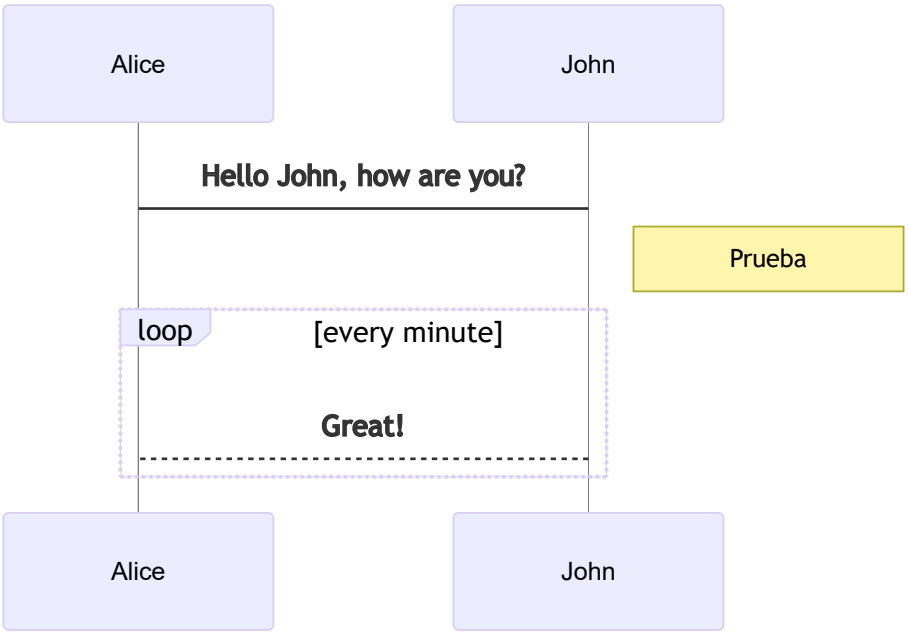
Pasaremos los datos necesarios para construir un objeto ISBN al constructor de Libro (en nuestro caso una cadena con el isbn) y será este quien lo cree. En el fondo esto es lo que sucede con un tipo valor como DateTime por su 'naturaleza'. De esta manera, además, el objeto Isbn13 que estoy creando desaparecerá junto con el libro.

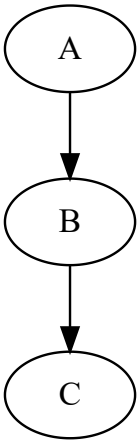
If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component: ♡

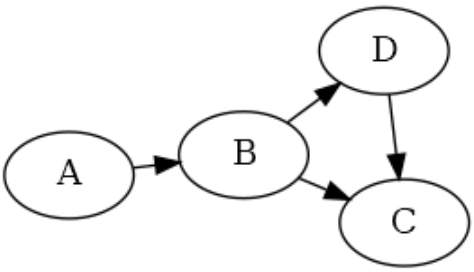
```java
# Post Here Code Example
static public void function base(int a, String b)
{
    System.out.println("hola");
}
```

```php
System.out.println("Hola: " + nombre);
```

```yaml
---
home: true
heroImage: /Logo_PSP.png
tagline: VuePress & Markdown
actionText: Access course →
actionLink: /unit1/
features:
- title: Feature 1 Título
  details: Feature 1 Description
- title: Feature 2 Title
  details: Feature 2 Description
- title: Feature 3 Title
  details: Feature 3 Description
footer: Made by Vicente Martínez with ♡
---
```







Hola
Cross-Relation

9.2. Browser API Access Restrictions

Because VuePress applications are server rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

9.2.1 Third level heading

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR- friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser

/ DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example **Java Title**

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

This is another component

```md
---
home: true
heroImage: /LogoPSP_en.png
tagline: Class notes
actionText: Access lessons →
actionLink: /unit1/
features:
- title:
  details:
- title:
```

```
  details:
- title:
  details:
footer: © 2021/2022
---

<div class="cclicense" align="center">
  <p xmlns:cc="http://creativecommons.org/ns#" xmlns:dct="http://purl.org/dc/terms/">Exce
</div>
```

Licencia

PSP class notes by Vicente Martínez is licensed under **CC BY-NC-SA 4.0**

Licencia

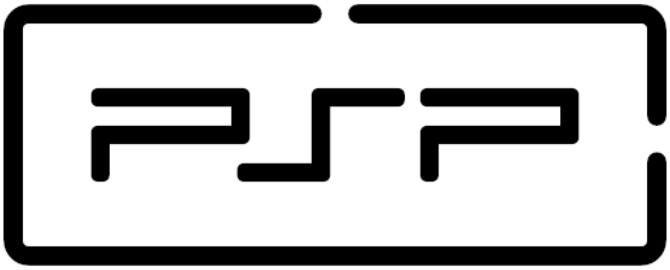
PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0

Licencia

PSP class notes by Vicente Martínez is licensed under **CC BY-NC-SA 4.0**

▶ Licencia

> **PSP class notes** by Vicente Martínez is licensed under **CC BY-NC-SA 4.0**



Process and Service Programming

Tema 1 - VuePress

Instalación y configuración

I.E.S.
Doctor Balmis

1. Instalación de VuePress v1.x

VuePress está formado por dos componente: un generador de sitios estáticos minimalista coun una configuración de temas basada en Vue y una API de extensiones , junto con un tema por defecto optimizado para la escritura de documentación técnica. VuePress fue inicialmente desarrollado para satisfacer las necesidades de documentación del desarrollo de Vue y sus subproyectos.

Vue utiliza el lenguaje MarkDown para generar el contenido. Cada página generada por VuePress se convierte en una renderización en HTML estático proporcionando rapidez de carga y permitiendo posibilidades de posicionamiento SEO. Sin embargo, durante la generación de las páginas, Vue coge el contenido estático y lo convierte en una `Single-Page Application (SPA)`. Con la configuración se puede generar una navegación entre las páginas del sitio, generéndose todos los enlaces y estructura el sitio de forma previa.

1.1 Instalación del entorno new!

Para poder usar VuePress, previamente debemos intalar Node.js en nuestro equipo.

A continuación, seguiremos los siguientes pasos para la instalación

```sh
mkdir vuepress-starter && cd vuepress-starter
npm init
npm install -D vuepress
mkdir docs && echo '# Hello VuePress' > docs/README.md
```

> **i Nota**
>
> En el último paso hemos creado una página de ejemplo con el título "Hello VuePress"

1.2 Lanzar el entorno de pruebas / producción

Una vez creadas las páginas, podemos ver en tiempo real como va quedando nuestro sitio y ver cómo se reflejan los cambios que vamos haciendo sobre el mismo.

La primera vez, tenemos que editar el archivo package.json y añadirle algunos scripts

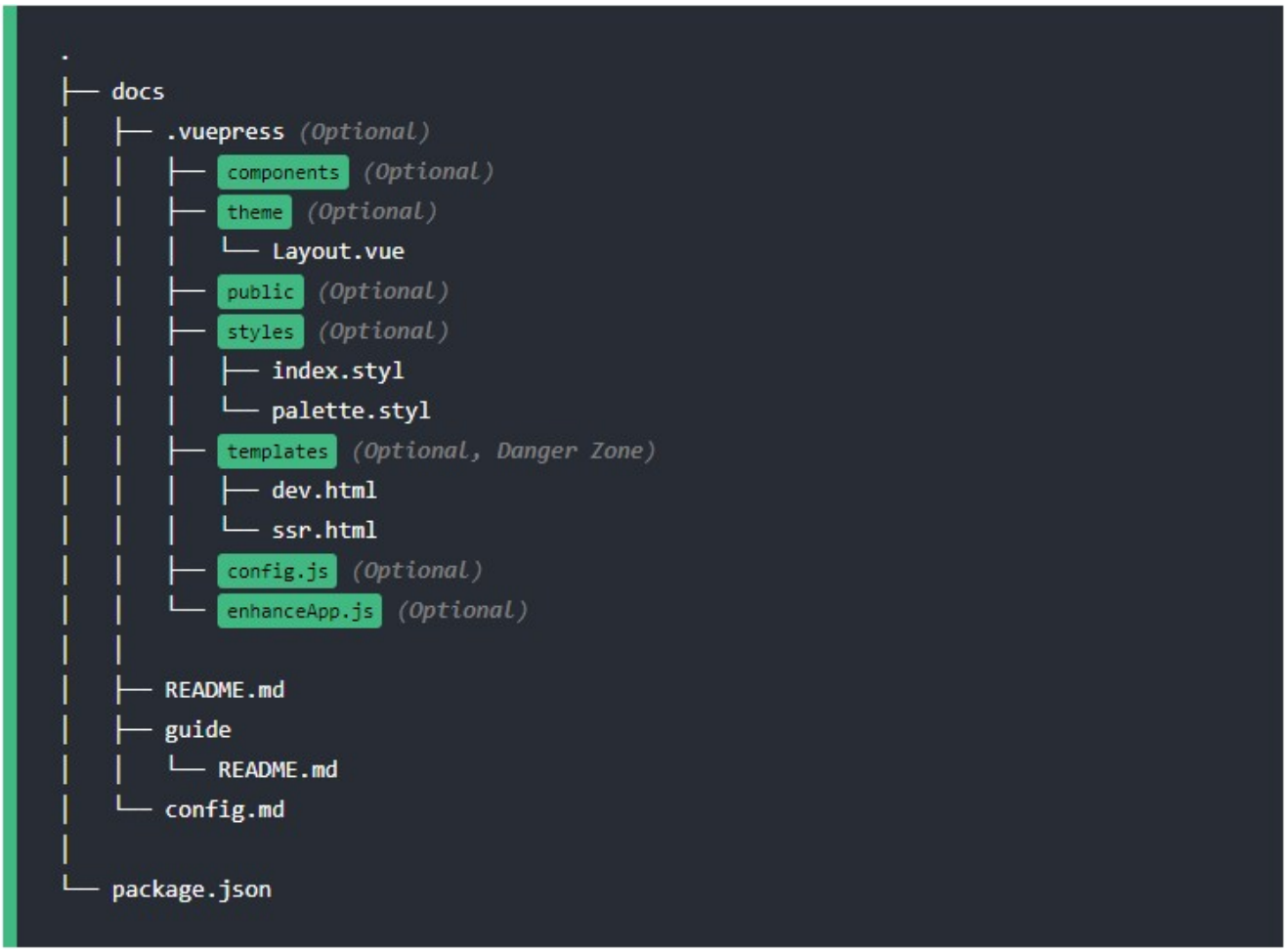
```json
"scripts": {
   "docs:dev": "vuepress dev docs",
   "docs:build": "vuepress build docs"
 }
```

A partir de ese momento podemos lanzar cualquiera de los dos entornos ejecutando los siguientes comandos desde el directorio raíz del proyecto (vuepress-starter en nuestro ejemplo)

```sh
# Lanzar entorno de desarrollo
npm run docs:dev
# Lanzar entorno de producción
npm run docs:build
```

> ⚠ **Diferencias**
>
> - El entorno de desarrollo lanza un servidor web local en localhost:8080 y va compilando los cambios sobre la marcha para que podamos visualizarlos en el servidor lanzado.
> - Por contra, el entrono de producción compila todos los archivos a HTML y genera una carpeta (docs/.vuepress/dist) con la exportación del sitio y todos los archivos necesarios para lanzarlo en cualquier servicio de alojamiento, bien sea GitHub Pages, un server en Azure o un servidor web local propio. En este proceso, se realizan comprobaciones de enlaces rotos, archivos inexistentes y errores en la configuración que harían que el sitio web generado no funcionase.

1.3 Estructura de directorios



- *.vuepress* es la carpeta que contiene todos los archivos de configuración y recursos estáticos (css, imágenes, pdf, ...) utilizados por el sitio web.
- *.vuepress/components* es la carpeta en la que podemos extender la funcionalidad por defecto de VuePress con nuevas etiquetas.

▶ 📋 Ejemplo de componente para generar la portada de los PDF (DocumentCover-component.vue)

- *.vuepress/styles/index.styl* es el archivo en el que podemos añadir o sobreescribir los estilos CSS del tema por defecto usado por VuePress. En mi caso yo lo utilizo principalmente para incluir/excluir estilos de la visualización en HTML y en PDF.

▶ 📋 Ejemplo de estilos aplicados

Un ejemplo de esta última ocnfiguración la podemos ver en los bloques de código que permite añadir VuePress para mostrar diferencias entre dos lenguajes o bien entre dos librerías del mismo lenguaje.

PHP Example **Java Title**

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

Versión pantalla vs imprimible

En la versión PDF de esta página, los códigos se mostrarán por separado. Esto se consigue con los estilos. Además, la portada se verá sólo en el PDF y los enlaces de descarga/visualización del PDF se verán solo en el HTML.

1.4 El archivo de configuración config.js

Sin modificar la configuración por defecto el sitio es mínimo y no hay navegación entre las páginas. Para configurar y adaptar el sitio a nuestro guesto, debemos crear el archivo .vuepress/config.js. En este archivo estará toda la configuración específica de VuePress.

En este archivo configuraremos:

- Características generales del sitio
- Configuración de los idiomas en los que se mostrarán las páginas
- Configuración de la barra de navegación superior
- Configuración de la barra de navegación lateral
- Configuración de los plugins que vayamos añadiendo a nuestro sitio
  - Copiar código desde las cajas
  - Ampliar imágenes
  - Extensiones de Markdown
  - Exportación PDF
  - Configuración del repositorio Github de publicación
  - ...

Hay infinidad de configuraciones y customizaciones que se pueden realizar a un sitio. Es la parte más costosa de la configuración de VuePress. Os dejo como ejemplo el archivo de configuración que tengo actualmente

```js
const { description } = require('../../package')

module.exports = {
  /**
   * Ref：https://v1.vuepress.vuejs.org/config/#title
   */
  title: 'PSP - 2º DAM',
  /**
   * Ref：https://v1.vuepress.vuejs.org/config/#description
   */
  description: 'Apuntes, PSP, DAM, FP, Programacion',

  /**
   * Directory to be deployed to http://mydeploymenthost/base/
   */
  // base: '/psp_sources/', // Producción
  base: '/', // Desarrollo

  // Host name for PDF export configuration
```

```
    host: 'localhost',

    /**
     * Extra tags to be injected to the page HTML `<head>`
     *
     * ref：https://v1.vuepress.vuejs.org/config/#head
     * Original theme-color: 3eaf7c
     */
    head: [
      ['meta', { name: 'theme-color', content: '#3e7caf' }],
      // original color ['meta', { name: 'theme-color', content: '#3eaf7c' }],
      ['meta', { name: 'apple-mobile-web-app-capable', content: 'yes' }],
      ['meta', { name: 'apple-mobile-web-app-status-bar-style', content: 'black' }],
      ['link', { rel: 'icon', href: '/psp_favicon.png' }]
    ],

    /**
     * Internationalization - Site Level i18n Config
     */

    locales: {
      // The key is the path for the locale to be nested under.
      // As a special case, the default locale can use '/' as its path.
      '/': {
        lang: 'en-US', // this will be set as the lang attribute on <html>
        title: 'Process and Service Programming',
        description: '2nd DAM PSP Module'
      },
      '/es/': {
        lang: 'es-ES',
        title: 'Programación de Servicios y Procesos',
        description: 'Módulo PSP de 2º DAM'
      }
    },

    theme: 'default-prefers-color-scheme',

    /**
     * Markdown format setting
     */
    markdown: {
      lineNumbers: true,
      anchor: {
        extractHeaders: [ 'h1', 'h2', 'h3' ],
        permalink: true,
        permalinkBefore: false,
        permaLinkAfter: true,
        permalinkSymbol: '&#x1F5F8;',
      },
      toc: {
```

```
      includeLevel: [1,2],
    },
  },

  /**
   * Theme configuration, here is the default theme configuration for VuePress.
   *
   * ref：https://v1.vuepress.vuejs.org/theme/default-theme-config.html
   */
  themeConfig: {

    // extended-theme-configurations
    overrideTheme: 'light',
    prefersTheme: 'light',

    // Sidebar behaviour
    sidebarDepth: 2,
    displayAllHeaders: true,

    locales: {
      '/': {
        // text for the language dropdown
        selectText: 'Languages',
        // label for this locale in the language dropdown
        label: 'English',
        // Aria Label for locale in the dropdown
        ariaLabel: 'Languages',
        // text for the edit-on-github link
        editLinkText: 'Help us improve this page on GitHub!',
        // config for Service Worker
        serviceWorker: {
          updatePopup: {
            message: "New content is available.",
            buttonText: "Refresh"
          }
        },
        nav: [
          { text: 'AULES', link: 'https://aules.edu.gva.es/fp/' , ariaLabel: 'AULES' },
          { text: 'Course Index', ariaLabel: 'Course Index',
            items: [
              { text: 'Unit 1 - Introduction to concurrent programming', link: '/unit1/'
              { text: 'Unit 2 - Processes programming', link: '/unit2/' }
            ]

          }
        ],
        sidebar: {
          '/unit1/': [ { title: 'PSP - Unit 1: Concurrent programming', collapsable: true
            children: ['', 'part1', 'part2', 'part1_']
          } ],
```

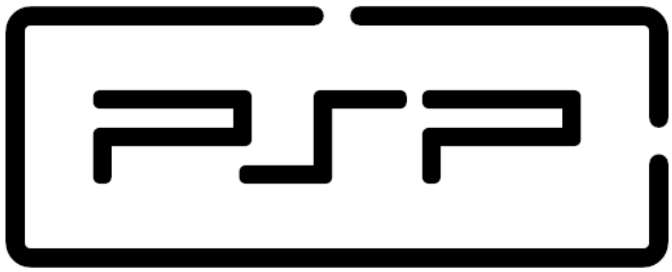
```
          '/unit2/': [ { title: 'PSP - Unit 2: Processes programming', collapsable: true,
            children: ['', 'part1', 'part2']
          } ],
          '/': [ { title: 'PSP - Contents', collapsable: true,
             children: ['', '/unit1/', '/unit2/' ]
          } ],
        }
      },
      '/es/': {
        // text for the language dropdown
        selectText: 'Idioma',
        // label for this locale in the language dropdown
        label: 'Castellano',
        // Aria Label for locale in the dropdown
        ariaLabel: 'Idioma',
        // text for the edit-on-github link
        editLinkText: 'Ayúdanos a mejorar esta página en GitHub!',
        // config for Service Worker
        serviceWorker: {
          updatePopup: {
            message: "Nuevo contenido disponible.",
            buttonText: "Actualizar"
          }
        },
        nav: [
          { text: 'AULES', link: 'https://aules.edu.gva.es/fp/' , ariaLabel: 'AULES' },
          { text: 'Contenido del curso', ariaLabel: 'Contenido del curso',
            items: [
              { text: 'Tema 1 - Introducción a la programación concurrente', link: '/es/u
              { text: 'Tema 2 - Programación de procesos', link: '/es/unit2/' }
            ]
          }
        ],
        sidebar: {
          '/es/unit1/': [ { title: 'PSP - Tema 1: Introducción a la programación concurre
            children: ['', 'part1', 'part2', 'part1_']
          } ],
          '/es/unit2/': [ { title: 'PSP - Tema 2: Programación de procesos', collapsable:
            children: ['', 'part1', 'part2']
          } ],
          '/es/': [ { title: 'PSP - Contenidos', collapsable: true,
            children: ['', '/unit1/', '/unit2/']
          } ],
        }
      }
    },
    // Smooth scrolling
    smoothScroll: true,
    // Github configuration
    // - Link to github repo in nav bar
```

```
    repo: 'psp2dam/psp_sources',
    repoLabel: 'GitHub',
    // - Link to page in github, at the bottom of each page
    docsDir: 'src',
    docsBranch: 'master',
    editLinks: true,
    editLinkText: 'Help us improve this page!',
    // Last updated info, at the bottom of each page
    lastUpdated: 'Last updated',
    // Navigation bar configuration, at top left on the nav bar (next to the title)
    logo: '/LogoIES.png',
  },

  /**
   * Apply plugins, ref：https://v1.vuepress.vuejs.org/zh/plugin/
   */
  plugins: {

    '@snowdog/vuepress-plugin-pdf-export': {
      puppeteerLaunchOptions: {
        args: ['--no-sandbox', '--disable-setuid-sandbox']
      },
      sorter: 'pathOrdered',
      outputFileName: 'book.pdf',
      theme: 'default-prefers-color-scheme',
      pageOptions: {
        format: 'A4',
        displayHeaderFooter: true,
        headerTemplate: `
        <style>
          .header, .footer{ font-family: "Segoe UI", Roboto, Oxygen, Arial, freesans, san
          .header { font-size: 10px; }
          .footer {font-size: 9px;}
          .border-bottom { border-bottom: 1pt solid #3eaf7c;}
          .border-top { border-top: 1pt solid #3eaf7c; border-right: 1pt solid #3eaf7c;}
        </style>
        <div class="border-bottom header"><span style="float:left;">CFGS DAM</span><span
      `,
        footerTemplate: `
        <div class="border-top footer">
          <span style="float:left; text-align:left;">IES Doctor Balmis</span>
          <b><span class="pageNumber"></span> / <span class="totalPages"></span></b>
        </div>
      `,
        printBackground: true,
        scale: 0.9,
        margin: {top: '1.5cm', bottom: '1.5cm', left: '1cm', right: '1cm', },
      },
    },
    '@vuepress/plugin-back-to-top': {
```

```
220
221       },
222       '@vuepress/plugin-medium-zoom': {
223         selector: 'p img, table img, figure img',
224         // medium-zoom options here
225         // See: https://github.com/francoischalifour/medium-zoom#options
226         options: {
227           margin: 48,
228           background: '#212530CC',
229           scrollOffset: 200,
230         }
231       },
232       '@vuepress/plugin-last-updated': {
233         dateOptions:{
234           hour12: false
235         }
236       },
237       '@xiaopanda/vuepress-plugin-code-copy': {
238         buttonStaticIcon: false,
239         buttonIconTitle: 'Copy',
240         buttonAlign: 'bottom',
241         buttonColor: '#3eaf7c'
242       },
243       'vuepress-plugin-mermaidjs': {
244
245       },
246       'vuepress-plugin-graphviz': {
247
248       }
249     }
250   }
251
```



Process and Service Programming

Unit 9.3 - Multiprogramming

Processes and threads

I.E.S.
Doctor Balmis

5. Using Vue in Markdown (print option)

This is a sample text to fill the paragraph in the documente

5.1. And now a paragraph

Unit 2. Processes programming

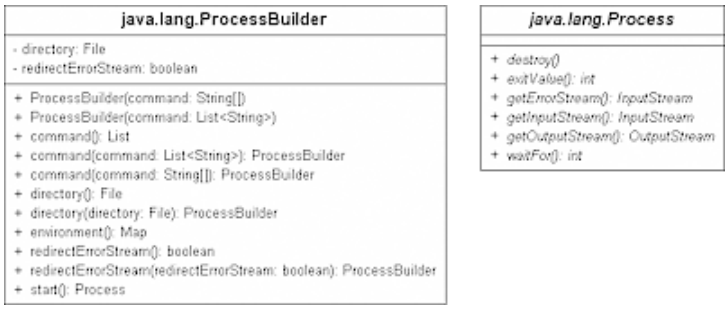
VuePress is composed of two parts: a **minimalistic static site generator** with a Vue-powered **theming system** and **Plugin API** , and a **default theme** optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

9. Using Vue in Markdown by Vicente default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

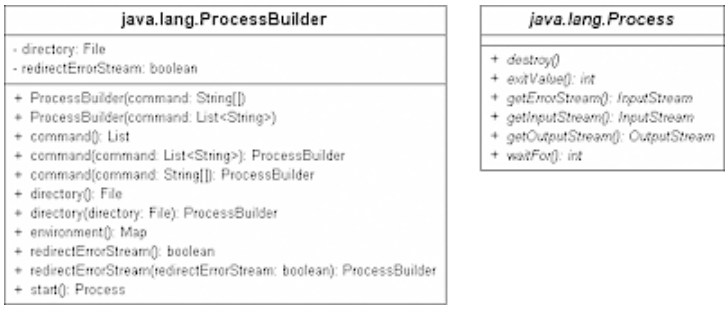
Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9. Using Vue in Markdown by Vicente default theme

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.



If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example Java Title

```php
# Post Here Code Example
```

```java
// Post Here Code Example
```

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

> **PRUEBA**
>
> This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the **universal code requirements** . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component: