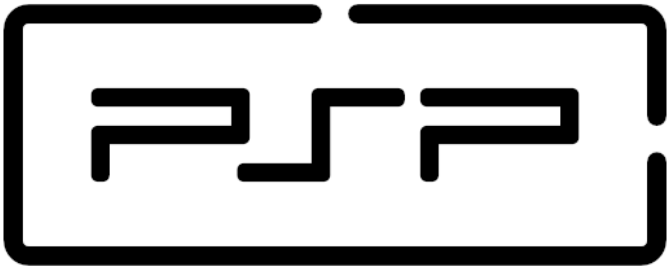


Programación de Servicios y Procesos

Tema 3 - Hilos

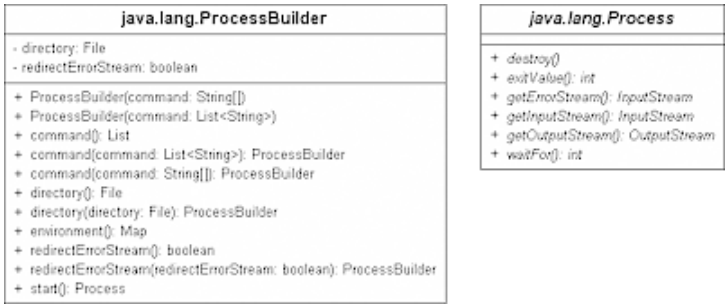
Threads y Runnable

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";
}
```

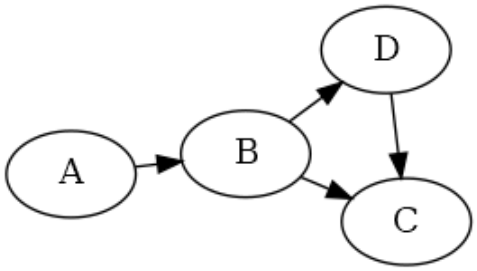
PHP Example **Java Title**

```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: