



Programación de Servicios y Procesos

Tema 3 - Hilos

Threads y Runnable



I.E.S.
Doctor Balmis

Apuntes de PSP creados por Vicente Martínez bajo licencia CC BY-NC-SA 4.0

9. Using Vue in Markdown by Vicente

default theme

- 9. Using Vue in Markdown by Vicente default theme
 - 9.1. Browser API Access Restrictions new!
 - 9.2. Browser API Access Restrictions
 - 9.3. Browser API Access Restrictions
 - 9.4. Browser API Access Restrictions
 - 9.5. Browser API Access Restrictions
 - 9.6. Browser API Access Restrictions
 - 9.7. Browser API Access Restrictions

9.1. Browser API Access Restrictions new!

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.

<code>java.lang.ProcessBuilder</code>	<code>java.lang.Process</code>
<ul style="list-style-type: none"> - <code>directory</code>: File - <code>redirectErrorStream</code>: boolean <ul style="list-style-type: none"> + <code>ProcessBuilder(command: String[])</code> + <code>ProcessBuilder(command: List<String>)</code> + <code>command(): List</code> + <code>command(command: List<String>): ProcessBuilder</code> + <code>command(command: String[]): ProcessBuilder</code> + <code>directory(): File</code> + <code>directory(directory: File): ProcessBuilder</code> + <code>environment(): Map</code> + <code>redirectErrorStream(): boolean</code> + <code>redirectErrorStream(redirectErrorStream: boolean): ProcessBuilder</code> + <code>start(): Process</code> 	<ul style="list-style-type: none"> + <code>destroy()</code> + <code>exitValue(): int</code> + <code>getErrorStream(): InputStream</code> + <code>getInputStream(): InputStream</code> + <code>getOutputStream(): OutputStream</code> + <code>waitFor(): int</code>

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: ❤

```

1  # Post Here Code Example
2  function base(a, b)
3  {
4      print "hola";
5  }
```

php

PHP Example **Java Title**

```
1 # Post Here Code Example
```

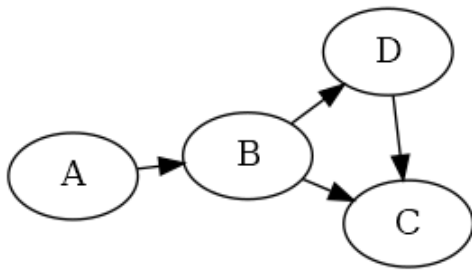
php

```
1 // Post Here Code Examples
```

java

```
1 ---
2 home: true
3 heroImage: /Logo_PSP.png
4 tagline: VuePress & Markdown
5 actionText: Access course →
6 actionLink: /unit1/
7 features:
8 - title: Feature 1 Título
9   details: Feature 1 Description
10 - title: Feature 2 Title
11   details: Feature 2 Description
12 - title: Feature 3 Title
13   details: Feature 3 Description
14 footer: Made by Vicente Martínez with ♥
15 ---
16
```

yaml

**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: