Introducción a GitHub Pages

Tiempo estimado: 10min

Introducción

Un sitio web estático (*static website*) es aquel que sólo contiene recursos que se sirven tal cual se encuentran almacenados en el servidor, sin ningún tipo adicional de procesamiento. No ejecutan código dinámico de ningún tipo como, por ejemplo, Node, PHP, Python o Perl. GitHub Pages es un servidor de sitios webs estáticos. Su página oficial es pages.github.com. Es un servicio que ofrece GitHub gratuitamente.

Actualmente, cerca de un millón de sitios webs se encuentran alojados en GitHub Pages como, por ejemplo, Bootstrap de Twitter, Electron de GitHub o React de Facebook.

Entre las principales ventajas de los sitios webs estáticos encontramos:

- Se sirven más rápidamente al no tener que ejecutar código dinámico.
- Son más seguros al no tener que ejecutar código dinámico.
- Suele ser más rápido el desarrollo de este tipo de sitios webs.
- Generalmente es más barato su alojamiento.

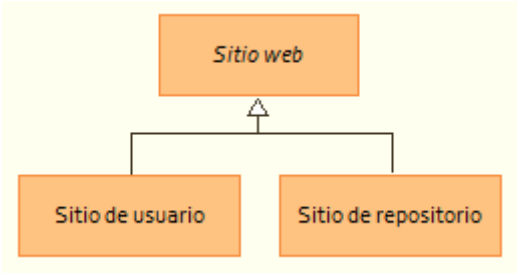
Características de GitHub Pages

Las principales características de GitHub Pages son:

- Es un servicio gratuito de GitHub.
- El contenido de los sitios webs se publica mediante Git. Por lo que unas nociones básicas de este sistema de control de versiones son indispensables.
- Es muy fácil de aprender.
- Es muy fácil de usar.
- Su contenido es siempre público, por lo que no se debe publicar sitios con datos privados en GitHub Pages.

Tipos de sitios webs

Para poder publicar un sitio web mediante GitHub Pages, necesitamos una cuenta de GitHub. Por otra parte, atendiendo al tipo de sitio web a publicar, GitHub Pages distingue dos tipos de sitios: los asociados a cuentas de usuario u organización o bien aquellos que están asociados a repositorios. Atendiendo a lo que deseemos, la publicación se hace de una manera u otra.



Un sitio de usuario (*user site*) o sitio de organización (*organization site*) es aquel que se asocia a una determinada cuenta de GitHub, ya sea de usuario o de organización. Su dominio será cuenta.github.io. Para cada cuenta, sólo puede haber un sitio web de este tipo.

Por su parte, un sitio de repositorio (*repository site*) o sitio de proyecto (*project site*) es aquel asociado

Copyright © 2016 nodemy.com. Reservados todos los derechos.

a un determinado repositorio. Como una misma cuenta puede tener varios repositorios, podríamos tener un sitio de este tipo para cada uno de ellos si fuera necesario. En este último caso, los dominios tienen el siguiente formato: cuenta.github.io/repositorio.

Lo primero que tenemos que tener en cuenta es cómo deseamos publicar nuestro sitio web en GitHub Pages. Sea uno asociado a cuenta o a repositorio. Una vez lo tenemos claro, lo siguiente es trabajar de una manera u otra. Veamos.

Sitio de usuario u organización

Recordemos que un sitio de usuario (*user site*) o sitio de organización (*organization site*) es aquel que se asocia a una cuenta. Cada cuenta puede tener como máximo un sitio web de este tipo. Su creación es muy, pero que muy sencilla: basta con crear un repositorio específico con el nombre cuenta.github.io. Así, por ejemplo, si nuestro nombre de cuenta es cuenta123, el repositorio a crear será cuenta123.github.io.

GitHub Pages publicará el contenido de la rama master de este repositorio. Así de simple.

Así pues, una vez creado el repositorio, bastará con clonarlo, modificarlo y subir las actualizaciones mediante el comando push de Git.

Clonación del repositorio

Una vez creado el repositorio en GitHub, lo siguiente es clonarlo para obtener nuestra copia local con la que trabajar. Para ello, usaremos el comando clone de Git:

```
git clone https://github.com/cuenta/cuenta.github.io.git
```

Una vez clonado, ya podemos entrar en el directorio local y hacer cambios.

Publicación de cambios

Una vez realizado los cambios en el contenido del sitio web, lo siguiente es publicarlos. Primero, recordemos, hay que registrar los cambios en el repositorio local:

```
git add .
```

A continuación, agrupar los cambios en una confirmación, para la cual debemos asociarle un nombre descriptivo:

```
git commit -m "descripción de los cambios"
```

Por ejemplo, si ya tenemos publicado el sitio web y, por cualquier razón, modificamos la imagen de logo, podríamos realizar la confirmación como sigue:

```
git commit -m "Actualización de logo"
```

Una vez confirmados los cambios, no hay más que subirlos al repositorio de GitHub:

```
git push
```

Hecho esto, GitHub Pages comenzará a servir el nuevo contenido. Esto no siempre es instantáneo. Algunas veces, la caché interna del servidor web sirve durante un pequeño período de tiempo la versión cacheada. Para resolver el problema, generalmente basta con ir al repositorio de GitHub, y solicitar explícitamente el archivo modificado que está siendo servido desde la caché. Esto suele bastar para resolver este pequeño problema.

Sitio de repositorio o proyecto

Si un sitio de usuario u organización se publica mediante su propio repositorio, un sitio de repositorio (*repository site*) o sitio de proyecto (*project site*) es aquel que se publica a través del propio repositorio del proyecto. Esta vez, el repositorio puede tener cualquier nombre, pero el contenido del sitio debe encontrarse en una rama especial, gh-pages. Todo lo que se encuentre en esta rama, será lo que publique GitHub Pages.

Tampoco es muy complicado, pero para los legos en la materia, la opción del sitio de usuario es mucho más fácil de entender y con la que trabajar.

Copyright © 2016 nodemy.com. Reservados todos los derechos.

Creación de la rama gh-pages

Recordemos que en Git una rama (*branch*) no es más que una parte del proyecto independiente, cuyos cambios se mantienen aislados del resto de ramas del repositorio. Por lo general, el repositorio mantiene una rama principal (*master branch*), aquella que, digamos, se considera la más importante y contiene el código principal del proyecto. En nuestro caso, hay que crear una rama independiente, gh-pages, cuyo contenido mantendremos siempre separado del resto de ramas.

Para comenzar, primero hay que crear la rama, por ejemplo, mediante el comando checkout de Git:

```
git checkout -b gh-pages
```

Una vez creada la rama, cada vez que deseemos saltar a ella y así modificarla y, por ende, modificar el contenido del sitio web estático publicado en GitHub Pages, tendremos que usar el comando checkout como sigue:

```
git checkout gh-pages
```

Publicación de cambios

La publicación de cambios es muy similar a la vista en los sitios de usuario u organización. Los comandos git add y git commit son los mismos y tienen los mismos objetivos. Registrar los cambios en el repositorio local. Ahora bien, su publicación muy similar, varía muy poco. El comando a usar es:

```
git push origin gh-pages
```

Con este comando lo que estamos haciendo es publicar los cambios de la rama gh-pages en el repositorio de GitHub y, por ende, en el servidor de GitHub Pages.

Recursos desconocidos

Cada vez que el usuario solicita un recurso que no existe en el sitio web, GitHub Pages responde con un mensaje HTTP con código 404. Si lo deseamos, podemos indicar el contenido que debe remitirse al usuario en estos casos. Para ello, tenemos que proporcionar un archivo 404.html ó 404.md en la raíz del proyecto. Cada vez que GitHub Pages remita el mensaje 404, en su cuerpo adjuntará el contenido de uno de estos archivos.

En muchas ocasiones, la mayoría, se desea remitir al usuario a la página índice del sitio. En este caso, puede utilizar el siguiente contenido como archivo 404.html:

```
<!DOCTYPE html>

<html>
  <body>
    <p>Redirigiendo a la raíz del sitio...</p>

    <script type="text/javascript">
      window.location.replace("/");
    </script>
  </body>
</html>
```

Lo que hace es muy simple. Cuando el usuario recibe el recurso, el navegador ejecuta la línea de código JavaScript que automáticamente remite a la raíz. La idea es que el usuario no llegue a ver el mensaje 404.

Dominio personalizado

Generalmente, no deseamos que los usuarios tengan que acceder a nuestro sitio a través del dominio de GitHub Pages, sea cuenta.github.io o cuenta.github.io/repositorio, según el tipo de sitio web publicado. Por suerte, GitHub Pages permite indicarle bajo que nombre de dominio deseamos sirva nuestro sitio. Para este objetivo, se encuentra el archivo CNAME, el cual debe encontrarse en la raíz del proyecto y debe contener una línea con nuestro nombre de dominio. Así, por ejemplo, si nuestro dominio fuese www.mi.com, el archivo CNAME deberá contener:

```
www.mi.com
```

Lo siguiente será actualizar la entrada DNS de www.mi.com, ajeno a GitHub Pages, para que apunte a la URL de nuestro sitio bajo GitHub Pages. Por ejemplo, si tenemos un dominio cuenta123.github.io al que deseamos acceder mediante www.mi.com, tendremos que crear una entrada CNAME en el servidor de

DNS cuyo contenido sea cuenta123.github.io. De esta manera, cuando los usuarios resuelvan www.mi.com, el servidor de DNS les remitirá a GitHub Pages.

A nuestro dominio particular, el que deseamos que utilicen los usuarios para acceder al sitio, GitHub Pages lo conoce formalmente como dominio personalizado (*custom domain*). Es importante conocerlo, de cara a la documentación con la que nos encontraremos.

Información de publicación

Título	Introducción a GitHub Pages
Autor	Raúl G. González - raulggonzalez@nodemy.com
Primera edición	Agosto de 2016
Última actualización	Octubre de 2016
Versión actual	1.0.1
Contacto	hola@nodemy.com

Copyright © 2016 nodemy.com. Reservados todos los derechos.