Estructura de carpetas en el código fuente

inicio de redacción: 09/feb/2022
últimos cambios: 10 de junio de 2022

Cada lenguaje tiene unas "mejores prácticas" y cada persona tiene sus propias preferencias personales. Así es que... cada cual suele organizarse como le apetece.

Esto es simplemente un resumen de ¿lo que suele ser más habitual?...

En C o C++

/ → directamente en la raiz suelen ir archivos tales como: CMakeLists.txt (makefile.txt,...), README.md, LICENSE.md,...

/bin/ → el programa ejecutable final resultante.

/build/ → los productos intermedios del proceso de compilacion/linkado.

/include/ → las cabeceras (.h) públicas.

/installer/ → el código fuente o los archivos de configuración necesarios para preparar el programa instalador del programa ejecutable final para hacerlo llegar a las máquinas de los usuarios.

/data/ → todo tipo de recursos necesarios para el funcionamiento del programa ejecutable final o de los test.

/doc/ → documentación aclaratoria, información interesante,...

/lib/ → bibliotecas propias o de 3ros que se estén utilizando.

/src/ → el código fuente (.h, .c, .cpp)

/test/ → el código fuente de los test (.h, .c, .cpp)

nota: No tienen por qué utilizarse todas y cada una de las carpetas citadas. Por ejemplo, si no se usan bibliotecas, /lib/ no seria necesaria; o si no hay cabeceras públicas, /include/ no seria necesaria; o si no hay instalador, /installer/ no seria necesaria; etc.
La idea es crear cada carpeta cuando la necesitemos.

nota: Si el proyecto es muy grande, es muy posible que se necesiten subcarpetas dentro de algunas de las carpetas principales que se han citado.

En java (Maven)

/ → directamente en la raiz suelen ir archivos tales como: pom.xml, README.md, LICENSE.md,...

/src/installer/ → el código fuente o los archivos de configuración necesarios para preparar el programa instalador del programa ejecutable final para hacerlo llegar a las máquinas de los usuarios.

/src/main/java/ → el código fuente.

/src/main/resources/ → todo tipo de recursos necesarios para el funcionamiento del programa ejecutable final.

/src/test/java/ → el código fuente de los test.

/src/test/resources/ → todo tipo de recursos necesarios para la ejecución de los test.

/target/ → el programa ejecutable final resultante.

nota: En java se suelen utilizar 'packages' para organizar el código y evitar colisiones de nombres (parecido a los 'namespaces' en otros lenguajes). Y la convención es bastante estricta: la estructura de los nombres de los paquetes ha de coincidir con la estructura de las carpetas en el disco.
Si tenemos un dominio distintivo propio (por ejemplo susosise.es), lo habitual es utilizar la convención de "url inversa": por ejemplo, el paquete con el módulo de almacén iría en la carpeta 'es', con la carpeta 'susosise' dentro de ella, con la carpeta 'almacen' dentro de ella → package 'es.susosise.almacen'.
En Maven quedaria así → /src/main/java/es/susosise/almacen/

En Python

/ → directamente en la raiz suelen ir archivos tales como: setup.py, README.md, LICENSE.md, requirements.txt,...

/nombredeproyecto/ → el código fuente de nuestro proyecto "nombredeproyecto".

Otras carpetas, si lo estimamos necesario.

En C#

Cuando se crea un proyecto en Visual Studio, el asistente genera un esqueleto base según el tipo de proyecto. Sobre él podemos ir construyendo. Eso sí, procurando seguir y respetar las convenciones plasmadas en él (si seguimos las convenciones propuestas por Microsoft, el trabajo será más fácil y nos ahorraremos problemas).

En un servidor web

/ → directamente en la raiz suelen ir archivos tales como: index.html (la página de inicio por defecto),. . . .

/styles → los archivos .css y otros recursos para aplicar estilos generales.

/images → fotos y gráficos generales.

/scripts → código javascript.

/lib → bibliotecas que necesiten los scripts.

/pages → las páginas del sitio web
nota: En webs grandes, en lugar de 'pages', irán diferentes carpetas; una para cada categoria contemplada. Suele ser importante que las carpetas tengan nombres descriptivos, para que las URLs resultantes sean semáticamente significativas.

La gente que prefiere nombres cortos, suele usar tres letras:
/
/css
/img
/js
/lib