Lanzar programa que ejecute un ping dos veces en Kotlin

[crayon-6a9d67637754b197582959/]

processBuilder.redirectOutput(ProcessBuilder.Redirect.INHERIT) : Esta línea configura la redirección de la salida estándar del proceso. ProcessBuilder.Redirect.INHERIT significa que la salida estándar del proceso se redirigirá a la salida estándar de la aplicación que lo ejecuta. En otras palabras, los resultados del comando «ping» se imprimirán en la consola donde se esté ejecutando el programa Kotlin.

Este código en Kotlin utiliza la clase ProcessBuilder para ejecutar comandos de ping en dos hosts diferentes (www.example.com y www.google.com) y muestra los resultados directamente en la consola. Aquí está la explicación línea por línea:

1. **Importación de la clase ProcessBuilder**: Se importa la clase ProcessBuilder de la biblioteca estándar de Java, que se utiliza para crear y controlar procesos del sistema operativo.
2. **Función principal main**: Se define la función main, que es el punto de entrada del programa.
3. **Definición de los hosts**: Se definen dos variables host1 y host2 que contienen las direcciones de los hosts a los que se quiere hacer ping.
4. **Llamadas a executePingCommand**: Se llama a la función executePingCommand dos veces, una para cada host, pasando las direcciones de los hosts como argumentos.
5. **Función executePingCommand**: Se define la función executePingCommand que acepta una cadena host como argumento y ejecuta el comando de ping en ese host.
6. **Creación del ProcessBuilder**: Se crea un objeto ProcessBuilder para ejecutar el comando ping con los

argumentos -c 5 y el nombre del host. El argumento -c 5 especifica que se deben enviar 5 paquetes de ping.

7. **Redirección de la salida y errores del proceso**: Se configuran las redirecciones para que tanto la salida estándar como la salida de errores del proceso se hereden del proceso padre (el programa Kotlin). Esto significa que los resultados del ping se mostrarán directamente en la consola.
8. **Inicio del proceso**: Se inicia el proceso utilizando el ProcessBuilder.
9. **Esperar a que el proceso termine**: Se espera a que el proceso termine antes de continuar. Esto asegura que el programa principal no continúe antes de que el proceso hijo haya completado su ejecución.