Programing Paradigms - Funcional

28-11-2022

Leandro Baier

Metas

Temas a revisar:

1. Conceptos (Paradigma Prog Funcional, Funciones puras, Composición de funciones, Mutabilidad, Estados Compartidos, Efecto Secundario, etc)
2. Ventajas & Desventajas de Prog. Funcional
3. Sintaxis Lenguajes de Programación Funcional (Kotling, Rust, Scala, Python, Javascript, otro)
4. Programación Funcional con 1 a 3 años de experiencia (en al menos uno de los lenguajes del nivel beginner)

1. Conceptos (Paradigma Prog Funcional, Funciones puras, Composición de funciones, Mutabilidad, Estados Compartidos, Efecto Secundario, etc)

La programación funcional es un paradigma declarativo. Nos enfocaremos en "qué" estamos haciendo y no en "cómo" se está haciendo que sería el enfoque imperativo. Esto quiere decir que nosotros expresaremos nuestra lógica sin describir controles de flujo; no usaremos ciclos o condicionales.

- Funciones puras: Las funciones puras, no son más que funciones, las cuales, dando el mismo input, siempre retornan el mismo output, además de no tener efectos secundarios.
- Composición de funciones: La composición de funciones es el proceso de combinar dos o más funciones, teniendo como finalidad ejecutar cada una de estas funciones en secuencia para obtener un resultado en concreto.
- Mutabilidad: Un objeto inmutable es aquel que no puede ser modificado una vez haya sido creado.
- Estados Compartidos: El estado compartido es cualquier variable, objeto o espacio de memoria que exista en un ámbito compartido. Un ámbito compartido puede incluir el alcance global o ámbitos de cierre.
- Efecto Secundario: Un efecto secundario es cualquier cambio de estado en la aplicación que sea observable fuera de la función llamada. Los efectos secundarios se evitan principalmente en la programación funcional, para tener como resultado programas mucho más fáciles de comprender y fáciles de probar.

2. Ventajas & Desventajas de Prog. Funcional

Ventajas:

- Los programas no tienen estados
- Muy adecuados para la paralelización
- El código se puede testear fácilmente
- Código fácilmente verificable, incluso las funciones sin estado se pueden verificar
- Fácil de combinar con la programación imperativa y orientada a objetos
- Código más preciso y más corto

Desventajas:

- Los datos (por ejemplo, las variables) no se pueden modificar
- No se permite el acceso eficiente a grandes cantidades de datos
- No se recomienda para conexiones a bases de datos y servidores
- No es adecuado para muchas recursiones de la misma pila
- La programación recurrente puede dar lugar a errores graves
- No apto para todas las tareas

3, 4. Sintaxis Lenguajes de Programación Funcional (Kotling, Rust, Scala, Python, Javascript, otro), Programación Funcional con 1 a 3 años de experiencia (en al menos uno de los lenguajes del nivel beginner)

Hace poco más de un año empecé a aprender Rust, dejo un link a una pequeña web app desarrollada con Rusy y Yew(Framework frontend compilado en wasm), para poder correrla es necesario tener instalado:

- Rust
- wasm-bindgen-cli (cargo install trunk wasm-bindgen-cli)
- trunk (cargo install --locked trunk)
- Toolchain wasm64-unknown-unknown (rustup target add wasm64-unknown-unknown)

Con todo instalado y con una consola abierta en la ruta raíz del proyecto (donde se encuentra el archivo Cargo.toml) ejecutar “trunk serve”, la aplicación se levanta en el puerto 8080.