Aplica principios de programación funcional

Edward Garson

La programación funcional ha experimentado recientemente un interés renovado por parte de la comunidad general de programadores. En parte porque las propiedades emergentes del paradigma funcional están bien posicionadas para abordar los desafíos que surgen con el cambio de nuestra industria para el procesamiento multi-core. Sin embargo, aunque ciertamente esta sea una importante aplicación, no es la razón por la cual este texto te incita a conocer a la programación funcional.

Dominar este paradigma puede mejorar considerablemente la calidad del código que uno escribe en otros contextos. Si comprendes profundamente y aplicas el paradigma, tus designs exhibirán un grado mucho mayor de *transparencia referencial*.

La *transparencia referencial* es una propriedad muy deseable: implica que las funciones consistentemente devuelven los mismos resultados dadas las mismas entradas, independientemente de dónde y cuándo fueron ejecutadas. Es decir, la evaluación de la función dependerá menos (idealmente nada) de los efectos colaterales de un estado mutable.

La principal causa de defectos en código imperativo puede ser rastreada hasta las variables mutables. Todos leyendo este texto hubieran investigado porque algún valor no era esperado en una situación particular. La visibilidad semántica (encapsulamiento) puede ayudar a mitigar estos defectos insidiosos o, por lo menos, reducir drásticamente su localización, pero el verdadero culpado son designs que emplean una excesiva mutabilidad.

Y ciertamente no recibimos mucha ayuda de la industria en este aspecto. Los cursos de introducción a la orientación a objetos propositalmente promoven un design lleno de mutabilidad. A menudo, los ejemplos

mostrados son compuestos por grafos de objetos de larga duración que constantemente llaman métodos "mutadores" entre sí. Esto pode ser peligroso, pero con una minuciosa practica de pruebas, particularmente asegurándose de "Simular comportamientos no objetos", se puede evitar la mutabilidad innecesario.

El saldo es un design que típicamente tiene una mejor asignación de responsabilidades entre numerosas y pequeñas funciones que actúan sobre argumentos pasados a ellas al revés de referencias a variables internas mutables. Habrá menos defectos y será mas fácil y sencillo depurarlos, ya que es más fácil localizar dónde valores erróneos fueron introducidos que deducir el contexto particular que resultó en una asignación incorrecta. Todo esto conduce a un grado mucho mayor de transparencia referencial y, positivamente, nada pode interiorizar estas ideas dentro de sus huesos como aprender un lenguaje de programación funcional, dónde este modelo es la norma no la excepción.

Es claro que este enfoque no es óptimo para todas las situaciones. Por ejemplo, en sistemas orientados a objetos, este estilo casi siempre ofrece mejores resultados en el desarrollo del modelo de dominio que en el desarrollo de la interfaz de usuario (donde las funciones sirven para partir la complejidad de las reglas de negocio).

Domina el paradigma de la programación funcional para que seas capaz de aplicar con criterio las lecciones aprendidas en otros ámbitos. Sus sistemas orientados a objetos (por ejemplo) resonarán con los beneficios de la transparencia referencial y estarán mucho más cercanos de sus contrapartes funcionales que tu possas imaginar. De hecho, algunos incluso dirían que el ápice de la programación funcional y de la orientación a objetos son simplemente un reflejo uno de otro, una forma de yin yang computacional.