Русский

Урок Спринт 3.2

Пример кода 1

Тестирование конкурентного кода включает в себя проверку корректности выполнения кода в условиях одновременного выполнения нескольких горутин. Рассмотрим примеры.

Тестирование конкурентного кода в Go с использованием `t.Parallel`

В этом примере мы рассмотрим тестирование конкурентного кода, который выполняет операции добавления и удаления записи из кеша. Мы используем функцию `t.Parallel` для запуска тестов параллельно.

```
// cache.go

package cache

import (
    "sync"
)

// Cache представляет собой простой кеш
type Cache struct {
    mu     sync.Mutex
    values map[string]string
}

// NewCache создает новый экземпляр кеша
func NewCache() *Cache {
    return &Cache{
        values: make(map[string]string),
    }
}

// Add добавляет запись в кеш
func (c *Cache) Add(key, value string) {
    c.mu.Lock()
    defer c.mu.Unlock()
    c.values[key] = value
}

// Delete удаляет запись из кеша
func (c *Cache) Delete(key string) {
    c.mu.Lock()
    defer c.mu.Unlock()
    delete(c.values, key)
}

// Get возвращает значение записи по ключу из кеша
func (c *Cache) Get(key string) string {
    c.mu.Lock()
    defer c.mu.Unlock()
    return c.values[key]
}
```

Теперь давайте напишем тест для этого кода с использованием `t.Parallel`.

```
// cache_test.go

package cache

import (
    "testing"
```

```
)

// TestConcurrentCache - тест для параллельного тестирования кеша
func TestConcurrentCache(t *testing.T) {
    // Создаем новый кеш
    cache := NewCache()

    // Добавляем запись в кеш (не в параллели)
    cache.Add("key1", "value1")

    // Запускаем тесты в параллели
    t.Run("AddRecord", func(t *testing.T) {
        t.Parallel()
        // Добавляем запись в кеш
        cache.Add("key2", "value2")
    })

    t.Run("DeleteRecord", func(t *testing.T) {
        t.Parallel()
        // Удаляем запись из кеша
        cache.Delete("key1")
    })

    // Проверяем наличие записи после всех операций
    t.Run("CheckRecord", func(t *testing.T) {
        t.Parallel()
        // Получаем значение записи по ключу
        value := cache.Get("key2")
        expectedValue := "value2"

        // Проверяем, что значение соответствует ожидаемому
        if value != expectedValue {
            t.Errorf("Expected value: %s, Actual value: %s", expectedValue, value)
        }
    })
}
```

В этом тесте мы создаём экземпляр кеша и выполняем несколько операций (добавление, удаление) параллельно. Используя `t.Parallel`, мы можем запускать эти тесты одновременно, что помогает обнаруживать проблемы в конкурентном коде.

Справка

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках сервиса, принадлежат АНО ДПО «Образовательные технологии Яндекса». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «Образовательные технологии Яндекса».
Пользовательское соглашение.

© 2018 – 2024 ООО «Яндекс»