

Из pytest в Go: тестовое окружение на фикстурах

Кулин Тимофей

Яндекс, YDB



Golang Conf
2023



Тестовое окружение на фикстурах

Буду говорить про:

- объём служебного кода
- примеры использования фикстур в pytest и go (fixenv)
- принятые решения

Тестовое окружение на фикстурах

Не буду говорить о:

- сравнении подходов к созданию окружений
- производительности кода
- методиках тестирования

Тест на Go, создание файла

```
func TestCreateFile_Go(t *testing.T) {  
    dir, err := os.MkdirTemp("", "")  
    if err != nil {  
        t.Fatalf("failed to create temp dir: %v", err)  
    }  
    defer func() {  
        _ = os.RemoveAll(dir)  
    }()  
    fpath := filepath.Join(dir, "temp")  
    f, err := os.Create(fpath)  
    if err != nil {  
        t.Fatalf("failed to create file: %v", err)  
    }  
    _ = f.Close()  
}
```

Тест на Go, создание папки

```
func TestCreateDir_Go(t *testing.T) {  
    dir, err := os.MkdirTemp("", "")  
    if err != nil {  
        t.Fatalf("failed to create temp dir: %v", err)  
    }  
    defer func() {  
        _ = os.RemoveAll(dir)  
    }()  
    fpath := filepath.Join(dir, "temp")  
    err = os.Mkdir(fpath, 0666)  
    if err != nil {  
        t.Fatalf("failed to create dir: %v", err)  
    }  
}
```

Тесты на Go, общая часть

```
func TestXXX(t *testing.T) {  
    dir, err := os.MkdirTemp("", "")  
    if err != nil {  
        t.Fatalf("failed to create temp dir: %v", err)  
    }  
    defer func() {  
        _ = os.RemoveAll(dir)  
    }()  
    ...  
}
```

The background of the slide is black with a complex pattern of thin, light blue wavy lines that create a sense of depth and movement, resembling a topographic map or a stylized forest floor.

Тем временем в pytest

Фикстуры в pytest

```
def test_create_file(folder):  
    path = path.join(folder, "tmp")  
    f = open(path, "wb")  
    f.close()
```

```
def test_create_dir(folder):  
    path = path.join(folder, "tmp")  
    os.mkdir(path)
```

```
@pytest.fixture()  
def folder():  
    dir = tempfile.mkdtemp()  
    yield dir  
    shutil.rmtree(dir)
```




Фикстуры

Фикстуры

- Меньше кода
- Ленивое создание окружения
- Внутри теста результат фикстуры всегда одинаковый
- Результат вызова фикстуры может переиспользоваться между тестами
- Сборка мусора в окружении

The background features a complex pattern of thin, light blue wavy lines that create a sense of depth and movement, resembling a topographic map or a stylized fingerprint. The lines are centered and radiate outwards, filling the entire frame.

Фіхену в действии

Тест создания файла с фикстурой

```
func TestCreateFile_Fixenv(t *testing.T) {  
    e := fixenv.New(t)  
    fpath := filepath.Join(Folder(e), "file")  
    f, err := os.Create(fpath)  
    if err != nil {  
        t.Fatalf("failed to create file: %v", err)  
    }  
    _ = f.Close()  
}
```

Тест создания папки с фикстурой

```
func TestCreateDir_Fixenv(t *testing.T) {  
    e := fixenv.New(t)  
    fpath := filepath.Join(Folder(e), "dir")  
    err := os.Mkdir(fpath, 0666)  
    if err != nil {  
        t.Fatalf("failed to create file: %v", err)  
    }  
}
```

Сравнение подходов

```
func Test_Go(t *testing.T) {  
    dir, err := os.MkdirTemp("", "")  
    if err != nil {  
        t.Fatalf("failed dir: %v", err)  
    }  
    defer func() {  
        _ = os.RemoveAll(dir)  
    }()  
}
```

```
fpath := filepath.Join(dir, "tmp")  
err = os.Mkdir(fpath, 0666)  
if err != nil {  
    t.Fatalf("failed: %v", err)  
}  
}
```

```
func Test_Fixenv(t *testing.T) {  
    e := fixenv.New(t)  
    fpath:=filepath.Join(Folder(e), "dir")  
    err := os.Mkdir(fpath, 0666)  
    if err != nil {  
        t.Fatalf("failed: %v", err)  
    }  
}
```

Пример фикстуры

```
func Folder(e fixenv.Env) string {
    f := func() (*fixenv.GenericResult[string], error) {
        dir, err := os.MkdirTemp("", "")
        if err != nil {
            return nil, err
        }
        clean := func() {
            _ = os.RemoveAll(dir)
        }
        return fixenv.NewGenericResultWithCleanup(dir, clean), nil
    }

    return fixenv.CacheResult(e, f)
}
```

Фикстуры, пример поинтересней

```
func(t *testing.T) {  
  
    t.Parallel()  
    e := New(t)  
    ctx := sf.Context(e)  
    requireErrorIs(t,  
        Accounts(e).TransferMoney(  
            ctx,  
            NamedAccountID(e, "alice"),  
            NamedAccountID(e, "bob"),  
            100,  
        ),  
        ErrNoMoney,  
    )  
  
}
```

- Запуск docker-контейнера с YDB
- Создание таблицы Accounts
- Создание счёта alice
- Создание счёта bob
- Попытка перевода денег
- Удаление счётов bob, alice
- ... другие тесты, если есть
- Удаление таблицы accounts
- Удаление docker-контейнера

<https://github.com/rekby/highload-2023>

The background features a complex pattern of thin, light blue wavy lines that create a sense of depth and movement, resembling a topographic map or a stylized fingerprint. The lines are centered and radiate outwards, filling the entire frame.

Фіхену изнутри

Как использовать в тестах?

- Имена параметров?
- Имена полей в структуре?
- Просто функции

id := AccountID(e)

Что оставить внутри фикстуры?

```
func NamedAccountID(e Env, name string) string {  
    ...  
}
```

- Работа с кешем?
- Обеспечение параллельности вызовов?
- Хранение взаимосвязей объектов?
- Придумывание идентификаторов?
- Логика создания объекта?
- Логика очистки объекта?

Что оставить внутри фикстуры?

```
func NamedAccountID(e Env, name string) string {  
    ...  
}
```

- Работа с кешем?
- Обеспечение параллельности вызовов?
- Хранение взаимосвязей объектов?
- Придумывание идентификаторов?
- Логика создания объекта?
- Логика очистки объекта?

Хранение состояния

- Переменная внутри теста?
- Глобальная переменная?

Глобальный кеш

```
type EnvT struct {  
    c *cache  
    ...  
}  
var (  
    globalCache *cache  
    ...  
)
```

→

- Локальное состояние теста ссылается на глобальный кеш
- Глобальность позволяет иметь общее состояние
- Локальный указатель позволяет запускать тесты на фикстуры независимо друг от друга

Как создать EnvT?

- Служебный код должен быть минимальным
- Нужна совместимость со стандартной библиотекой
- Нужна совместимость с бенчмарками
- Нужна совместимость со сторонними тестовыми библиотеками (моки, сьюты и т.п.)

Ответ: одна функция, принимающая подинтерфейс от testing.TB

Различие экземпляров фикстур

- Как обеспечить разные уровни кеширования?
- Как допустить параллельное исполнение из разных тестов?
- Как обеспечить очистку объектов, кешируемых между тестами?

Как различать фикстуры

```
key := struct {  
    Scope          CacheScope `json:"scope"`  
    ScopeName      string    `json:"scope_name"`  
    FunctionName   string    `json:"func"`  
    FileName       string    `json:"fname"`  
    Params         interface{} `json:"params"`  
}
```

...

```
keyBytes, err := json.Marshal(key)
```

...

Параллельное выполнение

- Параллельные вызовы из разных тестов (всё просто — разные ключи)
- Параллельные вызовы из одного теста (должен выполняться только один)
- Гарантия единственного вызова

```
type cache struct {  
    m          sync.RWMutex  
    store      map[cacheKey]cacheVal // значения  
    setLocks   map[cacheKey]*sync.Once // синхронизация  
}
```

Очистка глобального состояния

- Выполняется после всех тестов
- Сборщик мусора не поможет (нужно выполнять код)

```
func TestMain(m *testing.M) {  
    os.Exit(fixenv.RunTests(m))  
}
```

Виртуальный тест

- Единый интерфейс для фикстур
- Отсутствие специального кода внутри `fixenv`

```
type virtualTest struct
{
::
}
```

```
type T interface {
    Cleanup(...)
    FataIf(...)
    Logf(...)
    Name() string
    SkipNow()
    Skipped() bool
}
```

Защита от ошибок

- Защита от двойной инициализации в тесте
- Защита от использования глобального кеша без инициализации
- Защита от рекурсии в фикстурах (в планах)
- Защита от смешивания score'ов фикстур (в планах)

Голосуйте

- сокращайте служебный код в тестах
- чистите окружение после тестов
- используйте фикстуры — они упрощают тесты
- внесите свой вклад в разработку fixenv

<https://github.com/rekby/fixenv>

<https://github.com/rekby/highload-2023>



Тимофей Кулин

github.com/rekby



Golang Conf
2023