以下幾種情況需要資料庫環境

- 資料庫單元測試(某個存取資料庫的 function)，見範例: **modules/mysqlapi/rdb**
- **直接連到資料庫**模組的整合測試，見範例: **modules/mysqlapi/restful**
  - `query`, `f2e-api` 等模組

初始化

環境參數檢查

可由 `common/testing/flag` (**tFlag**) 來產生要跳過 Ginkgo 測試的前處理

```
var (
  // 產生 Skip 的訊息
  itSkipMessage = tFlag.OwlDbHelpString(tFlag.OWL_DB_GRAPH)
  // 產生 Ginkgo 用的 SkippingFactory
  itSkip = tFlag.BuildSkipFactoryOfOwlDb(tFlag.OWL_DB_GRAPH,
itSkipMessage)
)

// 會依環境的參數提供與否，決定是否跳過(skip)測試
var _ = Describe("xxx", itSkip.PrependBeforeEach(func() {
  /* Your test... */
}))
```

初始化

有 *DbFacade 物件的 package

從 Ginkgo 的 BeforeSuite 與 AfterSuite 初始化與釋放資料庫資源

範例說明:

- `tDb` - `common/testing/db`
- `tFlag` - `common/testing/flag`

```
var ginkgoDb = &tDb.GinkgoDb{}

// 初始化資料庫連線，若沒有環境參數，會產生 nil 值給 DBFacade
var _ = BeforeSuite(func() {
  DbFacade = ginkgoDb.InitDbFacadeByFlag(tFlag.OWL_DB_GRAPH)
})

// 釋放資料庫資料，DBFacade 是 nil 也沒有問題
var _ = AfterSuite(func() {
  ginkgoDb.ReleaseDbFacade(DbFacade)
  DbFacade = nil
})
```

無 *DbFacade 物件的 package

與前例相同，直接定義一個 ***DBFacade** 物件

```
var DbFacade *f.DbFacade
```

資料準備/移除

可建立 inTx 的 function，以減少程式碼

```
func inTx(sql ...string) {
  DbFacade.SqlDbCtrl.ExecQueriesInTx(sql...)
}
```

使用 BeforeEach 與 AfterEach 來『準備』與『移除』測試資料

```
var _ = Describe("Some db test...", itSkip.PrependBeforeEach(func() {
  BeforeEach(func() {
    inTx(
      `INSERT INTO ...`
    )
  })

  AfterEach(func() {
    inTx(
      `DELETE FROM ...`
    )
  })
})
```

若資料準備與移除只需一次，可參考Ginkgo Builder 來建立測試

參考文件

GoDoc

- `common/db/facade`
- `common/testing/db`
- `common/testing/flag`

外部函式

- Ginkgo
- Gomega

Last modified on 2017-10-06T17:35:39+08:00