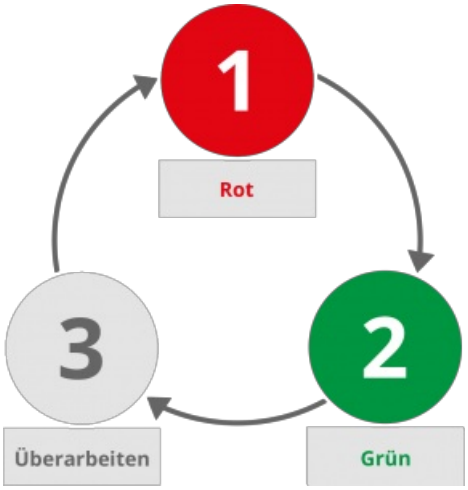
Softwareentwicklung für die pure.box

Testgetriebene Entwicklung von Go-Anwendungen

in der LiteIDE

Die LiteIDE ist eine freie, für alle Betriebssysteme verfügbare, Entwicklungsumgebung - ausgelegt auf die tägliche Arbeit mit Go. Sie unterstützt neben automatisierten Tests und dem Debuggen mit delve oder dem gdb auch die plattformübergreifende Kompilierung. Dieses Tutorial erklärt die testgetriebene Entwicklung von Go-Anwendungen. Dazu wird ein kleines Programm erstellt, das zwei Zahlen miteinander addiert.

Einleitung: Der testgetriebene Entwicklungszyklus



1. Rot: Alles beginnt mit einem fehlgeschlagenen Test:

Egal, ob ein neues Feature implementiert oder ein Bug gefixt werden soll: Der erste Schritt besteht stets darin, einen Test zu schreiben. Da noch kein Produktionscode existiert, schlägt er naturgemäß fehl.

2. Grün: Neuer Code erfüllt den Test:

Nun wird genau soviel Produktionscode geschrieben, wie benötigt wird, um den Test erfolgreich auszuführen. Der Code muss nicht besonders schön sein, aber eine möglichst unkomplizierte Lösung für das Teilproblem bereitstellen.

3. Überarbeiten: Lesbarkeit und Wartbarkeit sicherstellen

Im letzten Schritt wird der Produktionscode aufgeräumt, überarbeitet und ausführlich kommentiert. Hierbei geht es darum, den entstandenen Code lesbar und wartbar zu halten.

Nach mehreren Iterationen in diesem Zyklus sind umfangreiche Testsammlungen entstanden. Bei der Entwicklung neuer Features werden alle Tests ausgeführt. Auf diese Art und Weise wird ein enger Rahmen für neuen Programmteile vorgegeben, was im Allgemeinen zu einer hohen Codequalität führt.

Vorbereitung: Installation der Entwicklungsumgebung

Haben Sie go bereits installiert? Wenn nicht, arbeiten Sie zunächst bitte das Tutorial: "Erste Schritte in Go" (Windows) (Linux) durch.

Laden Sie die LiteIDE für Ihr Betriebssystem von Sourceforge herunter oder installieren Sie sie über ihren Paketmanager. Die LiteIDE kann in ein beliebiges Verzeichnis entpackt und von dort aus gestartet werden. Obwohl sie bereits alle notwendigen Tools mitbringt, empfiehlt sich die Installation weiterer Programme und Bibliotheken, die die Funktionalität erweitern:

Guru

```
go get -u golang.org/x/tools/guru
go install golang.org/x/tools/guru
```

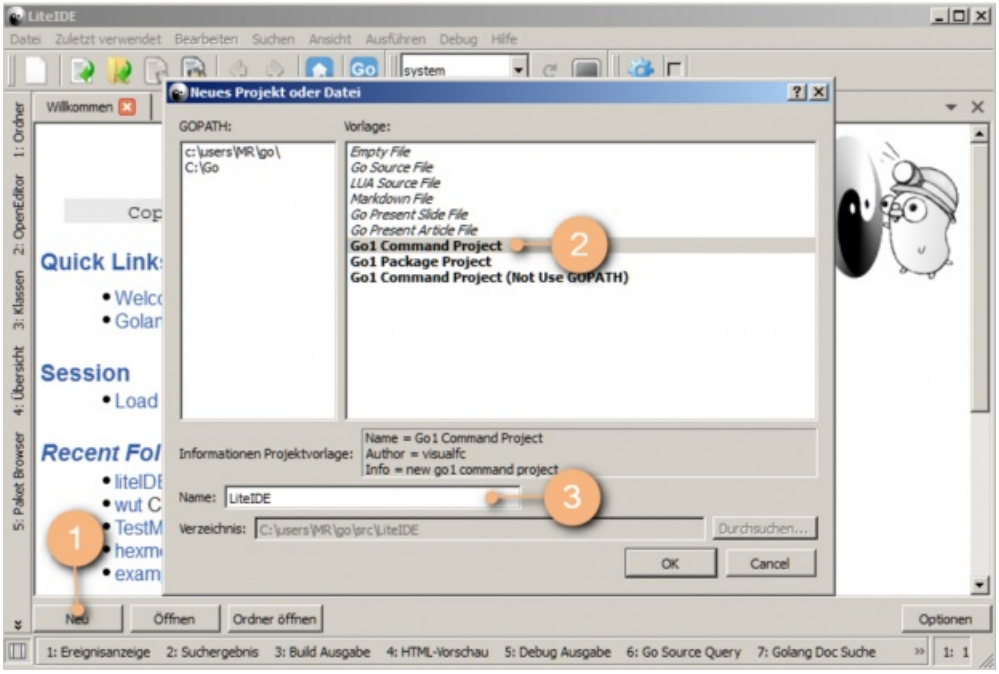
Golint

```
get -u github.com/golang/lint/golint
go install github.com/golang/lint/golint
```

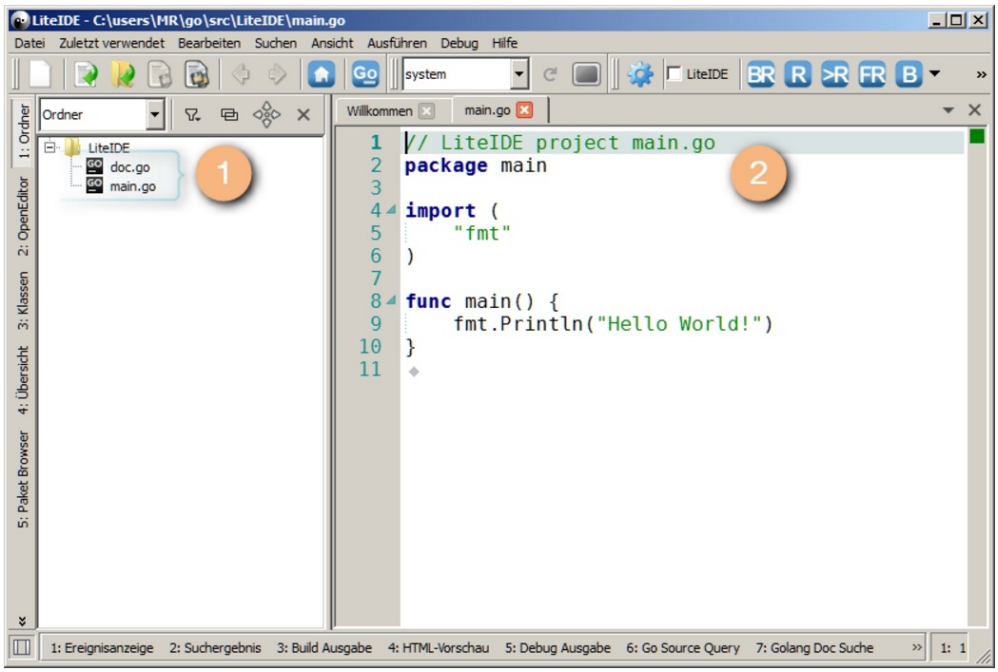
Delve

```
go get -u github.com/derekparker/delve
go install github.com/derekparker/delve
```

Projekt anlegen



1. (1) Klicken Sie auf die Schaltfläche "Neu" oder wählen Sie aus dem Menü "Datei" den Punkt "Neu".
2. (2) Wählen Sie "Go1 Command Project"
3. (3) Geben Sie dem Projekt einen Namen, beispielsweise "litetutorial"



Das neu angelegte Projekt enthält bereits zwei Dateien: Die Datei **go.doc** (1), die die Projektdokumentation enthalten soll, sowie die Editorfenster geöffnete Datei **main.go** (2), in die die IDE ein Hello-World-Programm geschrieben hat.

1. Rot: Test erstellen und ausführen

Dem Grundsatz “Keine Zeile Code ohne einen fehlgeschlagenen Test” folgend, wird im ersten Schritt ein Test für die Funktion Add(a,b) erstellt. Wird das Go-Tool mit dem Kommando go test aufgerufen, sucht es in Dateien, die auf _test.go enden, nach Funktionen, die mit einem großgeschriebenen Test beginnen.

Legen Sie also die Datei litetutorial_test.go an

```
package main

import "testing"

func TestAdd(t *testing.T) {
  a,b := 2, 3
  want := 5
  result := Add(a, b)

  if result != want {
    t.Errorf("Error in Addition: %d + %d = %d ", a, b, result)
  }
}
```

Zeile 1:
Sie wollen eine ausführbare Datei erstellen. Daher ist das Paket, in dem Sie arbeiten, das Paket main.

Zeile 3:
Um einen Test zu schreiben, verwenden Sie das Paket `testing` aus der Standardbibliothek.

Zeile 5:
Eine Funktion, die mit Test beginnt, wird als Test ausgeführt. Als Parameter bekommt die Funktion einen Pointer auf einen `testing.T`-Objekt übergeben.

Zeile 11:
Wenn die die Funktion `t.Errorf()` aufgerufen wird, schlägt der Test fehl. `t.Errorf()` funktioniert ähnlich wie die Funktion `fmt.Printf()` oder C-Funktion `printf()` und bekommt einen Formatstring, sowie verschiedene Werte für die Ausgabe übergeben.

Über die Schaltfläche “T” am oberen Bildschirmrand - oder über die Tastenkombination Strg+T wird der Test gestartet. Da die Funktion `Add(a,b)` nicht existiert, schlägt der Test fehl.

```
C:/Go/bin/go.exe test -v [C:/Users/Gopher/go/src/litetutorial]
=== RUN TestAdd
--- FAIL: TestAdd (0.00s)
  liteIDE_test.go:12: Error in Addition: 2 + 3 = -1
  FAIL
  exit status 1
  FAIL litetutorial 0.019s
  Fehler: Prozess beendet mit Rückgabewert 1.
```

2. Grün: Produktionscode implementieren und erneut Testen

Da der Test fehlgeschlagen ist, kann jetzt die Funktion `Add(a,b)` implementiert werden. Um die Codebasis zu strukturieren, wird die Funktion in in die Datei `calcfunctions.go` aus ausgelagert.

```
package main
func Add(a, b int) int {
  var result in
  result = a + b
  return result
}
```

Nach einem weiteren Aufruf des Test über die Tastenkombination Strg+T läuft der Test erfolgreich durch.

3. Aufräumen

Die letzte Phase des testgetriebenen Entwicklungszyklus wird gerne unterschätzt: Das Aufräumen. In dieser Phase wird der Code überarbeitet, so dass er am Ende gut verständlich und wartbar wird. Der Programmcode wird also auf das

Wesentliche reduziert und kommentiert:

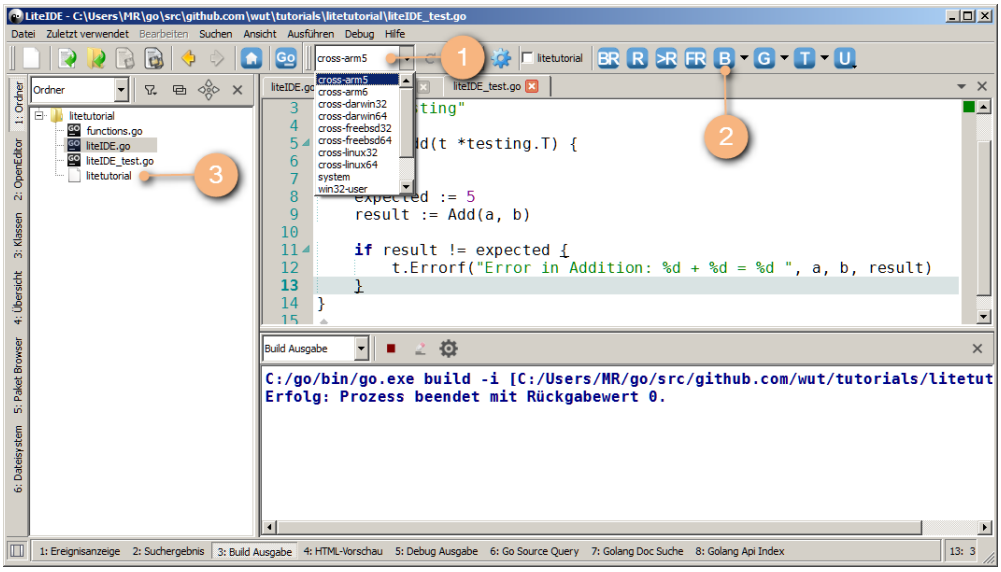
```
//Add returns the sum of to integers a and b
func Add(a, b int) int {
 return a+b
}
```

Ein abermals durchgeführter Test läuft ohne Fehler durch. Die Funktion kann jetzt im Hauptprogramm `litetutorial.go` verwendet werden.

```
package main

import "fmt"

func main() {
 a := 2
 b := 3
 sum := Add(a, b)

 fmt.Println(a, "+", b, "=", sum, "\n")
}
```

Mit einem Test für die Subtraktion kann nun ein neuer Entwicklungszyklus beginnen.

Programm für die pure.box kompilieren



1 Wählen Sie aus der System-Auswahl "cross-arm5"

2 Durch einen Klick auf das B-Symbol kompilieren Sie das Programm

3 Anschließend liegt im Projektverzeichnis die ausführbare Datei.

Wie im Tutorial "Erste Schritte in Go" behandelt, kann das fertige Binary jetzt per FTP, Samba oder SCP auf die pure.box geladen und ausgeführt werden.

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17 Uhr)
Fax: 0202/2680-265
info@wut.de