Verteilte Softwaresysteme

Blatt 1

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 09.10.2018 12:41

Gemeinsames Repository

Es gibt ein gemeinsames Repository für die Veranstaltung. Damit ich weiß wer von Ihnen am Praktikum teilnimmt um einen Schein zu bekommen, gibt es zwei Möglichkeiten darauf zuzugreifen:

1. Wenn Sie **keinen** Schein benötigen, greifen Sie einfach direkt auf das Repository zu.
2. Wenn Sie einen Schein bekommen wollen, treten Sie über den GitHub-Classroom-Link `https://classroom.github.com/g/RwBeRXrB` dem **einzigen** Team `ws18` bei. Erzeugen Sie **auf keinen Fall** ein anderes Team. Wählen Sie dabei Ihre E-Mail-Adresse aus, damit ich Ihnen Ihren GitHub-Account richtig zuordnen kann. Wenn Ihre E-Mail-Adresse nicht auftaucht, heisst das Sie sind im ZPA nicht für diese Veranstaltung eingetragen und können keine Leistungen ablegen.

Tools

Auf den Laborrechnern ist bereits die Go und GoLand installiert. Wenn Sie den Atom-Editor bevorzugen, empfiehlt es sich noch einige Tools zu installieren, wie z.B. hier beschrieben. Nicht auf den Laborrechner installiert ist der Visual Studio Code, für den es auch eine Go-Extension gibt.

GOPATH auf den Laborrechnern

Der GOPATH ist auf den Laborrechnern standardmäßig auf `C:Users\hm- ... \go` gesetzt. Damit das Roaming auf den MWN-PCs funktioniert, sollte aber alles besser auf `H:`

liegen. Gehen Sie zum Ändern des GOPATH auf den Laborrechnern wie unter `https://github.com/golang/go/wiki/SettingGOPATH#windows` beschrieben vor. Erzeugen Sie im ersten Schritt beispielsweise das Verzeichnis `H:\Benutzer\go`. Gehen Sie dann auf die Bearbeitung der Umgebungsvariablen für Ihr Konto.

Einarbeitung in Go

Über die Go-Website finden Sie eine Menge hilfreicher Ressourcen um sich in Go einzuarbeiten. Einen ersten *Berieselungseindruck* bekommen Sie beispielsweise durch das Video von Russ Cox auf YouTube. Eine sehr gute Möglichkeit für erste eigene Schritte in Go bietet die Tour of Go, die Sie im Browser, inkl. dem Ausführen von Go-Code, durchlaufen können.

Aufgabe

Unter dem Link `https://classroom.github.com/a/ujismuYm` bekommen Sie ein Git-Repository auf GitHub in dem Sie diese Aufgabe bearbeiten können. In dem Repository finden Sie bereits das Verzeichnis `moviestore` in dem die zwei Test-Dateien `movie_test.go` und `moviestore_test.go` mit ein paar wenigen Tests enthalten sind. In diesem Verzeichnis müssen Sie die weiteren Go-Dateien hinzufügen, damit sie zum Package `moviestore` gehören.

Ihre Aufgabe ist es den Quellcode für das Package `moviestore` zu schreiben und die Tests sinnvoll zu ergänzen. Immer wenn Sie in Ihr GitHub-Repository pushen, wird auf Travis CI ein Build getriggert. Was genau auf dem Travis CI gemacht wird, steuert die im Repository enthaltene Datei `.travis.yml`.

Committen Sie oft und in kleinen "Häppchen". Pushen Sie jedesmal, damit der Travis Sie aktiv bei der Entwicklung unterstützen kann. Sie können die Tools natürlich auch lokal ausführen. Installieren Sie dazu einfach GolangCI-Lint, wie dort in der README beschrieben.

Package Documentation

Die generierte Dokumentation für das `moviestore`-Package, sieht wie folgt aus. Die Implementierungen sollten Sie sinnvoll auf die Dateien `movie.go`, `user.go` und `moviestore.go` verteilen.

```
PACKAGE DOCUMENTATION

package moviestore
    import "github.com/ ... /moviestore"
```

```
TYPES

type Age uint8
    Age is just an unsigned 8-bit int, we do not expect older users.

type FSK uint8
    FSK is an unsigned 8-bit int

const (
    FSK0  FSK = 0
    FSK6  FSK = 6
    FSK12 FSK = 12
    FSK16 FSK = 16
    FSK18 FSK = 18
)
    See
    https://de.wikipedia.org/wiki/Freiwillige_Selbstkontrolle_der_Filmwirtschaft

type Movie struct {
    Title  string
    Fsk    FSK
    Serial Serial
}

    A Movie consists of a title, the fsk minimum age and a serial

func AllowedAtAge(m *Movie, age Age) bool
    AllowedAtAge checks whether the movie is allowed at a given age or not.

func (m *Movie) String() string
    Returns a string representing the movie like that:

    ”  23. Texas Chainsaw Massacre (Ab 18)”

    The serial field should be 4 digits wide.

type Moviestore interface {
    AddMovie(string, FSK) Serial
    AddUser(string, Age) UserID
    Rent(Serial, UserID) (User, Movie, error)
    Return(Serial) (User, Movie, error)
    RentedByUser(UserID) ([]Movie, error)
```

```
}
    A Moviestore interface.

func NewMoviestore() Moviestore
    NewMoviestore generates a Moviestore which contains available movies,
    users and a mapping between users and currently rented movies.

type Serial uint
    Serial is an unsigned int

type User struct {
    Name   string
    Age    Age
    UserID UserID
}

    User with name, age and id.

func (u *User) String() string
    Returns a string representing the use like that:

    "   8. Helga Meier, 28"

    The userid field should be 4 digits wide.

type UserID uint16
    UserID is just an unsigned 16-bit int.

type moviestoreImpl struct {
    available  map[Serial]Movie
    rented     map[UserID][]Movie
    users      map[UserID]User
    nextSerial Serial
    nextUserID UserID
}


func (ms *moviestoreImpl) AddMovie(title string, fsk FSK) Serial
    AddMovie adds a movie to the available movies map which is part of the
    moviestoreImpl struct:

    available   map[Serial]Movie

    The serial will be generated and returned.
```

```
func (ms *moviestoreImpl) AddUser(name string, age Age) UserID
    AddUser adds an user to the users map which is part of the
    moviestoreImpl struct:

    users   map[UserID]User

    The userid will be generated and returned.

func (ms *moviestoreImpl) Rent(serial Serial, userID UserID) (User, Movie, error)
    Rent a movie. If the user is in users and the movie is in available, the
    movie will be removed from available and appended to the slice of rented
    movies by this user. Therefore, the moviestoreImpl struct contains a
    field

    rented    map[UserID][]Movie

    The following error cases are handled and will be returned as error
    containing the following texts:
    - user not found
    - movie not available for rent
    - user ist too young

func (ms *moviestoreImpl) RentedByUser(userID UserID) ([]Movie, error)
    RentedByUser returns a slice of movies currently rented by the user.
    Error case is "userID unknown" Be aware that slices are returned by
    reference.

func (ms *moviestoreImpl) Return(serial Serial) (User, Movie, error)
    Return a movie. Searches in all slices of the rented map, does some
    "housekeeping, and returns the user and movie if found. The error case
    is "movie not found in rented movies".
```

Implementieren Sie, schreiben Sie Tests, lernen Sie Go und fragen Sie mich.