

COS 316

Precept:

Reflection

Overview of Reflection

- What is reflection?
 - Ability of program to examine / introspect itself during runtime
 - Sometimes called “meta-programming”
- Examples
 - Java Reflection API: discover methods and attributes at run-time, create objects of classes, whose names discovered only at run-time
 - C++ Runtime type identification (RTTI)

Java Reflection - Example

```
public class Stopwatch {  
  
    private long start;  
  
    public Stopwatch() {  
        start = System.currentTimeMillis();  
    }  
  
    public double elapsedTime() {  
        long now = System.currentTimeMillis();  
        return (now - start) / 1000.0;  
    }  
}
```

```
Stopwatch w1 = new Stopwatch();  
  
...  
  
System.out.println(w1.elapsedTime());
```

```
import java.lang.reflect.Constructor;  
import java.lang.reflect.Method;  
  
Class<?> swClass = Class.forName("Stopwatch");  
Constructor<?> c = swClass.getConstructor();  
Method[] methods = swClass.getMethods();  
Object w2 = c.newInstance();  
...  
System.out.println(methods[0].invoke(w2));
```

Go Reflection - Simple Example

```
package main
```

```
import (  
    "fmt"  
)
```

```
func main() {  
    i := 10  
    fmt.Printf("%v %T", i, i)  
}
```

- Try this example:

<https://go.dev/play/p/useMsla00m1>

- Modify and try with other types for i
 - What happens?

Go Reflection - MovieLens

```
package main
```

```
import (  
    "fmt"  
)
```

```
type movie struct {  
    movieId int    // represents the movie id  
    title string  // title of movie  
    year int      // movie release year  
    genre string  // pipe-separated list of genres  
}
```

```
func main() {  
    joker := movie {movieId: 193612, title:"Joker",  
                    year: 2019, genre: "action|thriller"}  
    fmt.Println(joker)  
}
```

- Try this example:

<https://play.golang.org/p/Int3mxYzt4n>

Go Reflection - MovieLens SQL Query

```
insert into Movies values(193612, "Joker", 2019, "action|thriller")
```

```
package main

import (
    "fmt"
)

type movie struct {
    movieId int    // represents the movie id
    title string  // title of movie
    year int     // movie release year
    genre string  // pipe-separated list of genres
}

func createQuery(m movie) string {
    q := fmt.Sprintf("insert into Movies values(%d, %s, %d, %s)",
        m.movieId, m.title, m.year, m.genre)

    return q
}

func main() {
    joker := movie {movieId: 193612, title:"Joker",
        year: 2019, genre: "action|thriller"}
    fmt.Println(createQuery(joker))
}
```

- Try this example:

<https://play.golang.org/p/KHb30DwBI fz>

- createQuery works for the movie struct but would it work for a different struct?

Go Reflection - MovieLens SQL Query

Question – write a general function that outputs a query for ANY struct?


```
type movie struct {  
    movieId int    // represents the movie id  
    title string  // title of movie  
    year int      // movie release year  
    genre string  // pipe-separated list of genres  
}
```

```
type links struct {  
    movieId int    // represents the movie id  
    imdbId int     // IMDB code  
    tmdbId int     // TMDB code  
}
```

```
jokerMovie := movie {movieId: 193612, title:"Joker",  
                      year: 2019, genre: "action|thriller"}
```


```
jokerLinks := links {movieId: 193612, imdbId: 7286456,  
                      tmdbId: 475557}
```

```
q1 := createQuery(jokerMovie)  
fmt.Println(q1)
```



```
insert into Movies values(193612, "Joker", 2019, "action|thriller")
```

```
q2 := createQuery(jokerLinks)  
fmt.Println(q2)
```



```
insert into Links values(193612, 7286456, 475557)
```

Go Reflection - createQuery & Reflection

- How can the createQuery function operate on any struct?
 - Use the empty interface{} argument type
 - Every type satisfies the empty interface: interface{}
- createQuery function must be able to
 - examine the type of the struct argument passed to it at run-time
 - discover its fields
 - create the query

Go Reflection - the Reflect package

- <https://golang.org/pkg/reflect>
- Provides tools:
 - identify the underlying concrete ***type*** of an `interface{}` variable
 - identify the ***value*** of an `interface{}` variable

Go Reflection -

reflect.TypeOf, reflect.Kind, reflect.ValueOf

```
package main
```

```
import (
```

```
    "fmt"
```

```
    "reflect")
```

```
type movie struct {
```

```
    movieId int    // represents the movie id
```

```
    title string  // title of movie
```

```
    year int      // movie release year
```

```
    genre string  // pipe-separated list of genres}
```

```
}
```

```
func createQuery(q interface{}) {
```

```
    t := reflect.TypeOf(q)
```

```
    k := t.Kind()
```

```
    v := reflect.ValueOf(q)
```

```
    fmt.Println("TypeOf ", t)
```

```
    fmt.Println("Kind   ", k)
```

```
    fmt.Println("ValueOf ", v)
```

```
}
```

```
func main() {
```

```
    joker := movie {movieId: 193612, title:"Joker",  
                    year: 2019, genre: "action|thriller"}
```

```
    createQuery(joker)
```

```
}
```

- Try this example:

<https://play.golang.org/p/8azcZB2yRfq>

- Extend this example with links struct

```
type links struct {
```

```
    movieId int    // represents the movie id
```

```
    imdbId  int     // IMDB code
```

```
    tmdbId  int     // TMDB code
```

```
}
```

Go Reflection -

reflect.NumFields, reflect.Fields

```
package main
import (
    "fmt"
    "reflect")

type movie struct {
    movieId int    // represents the movie id
    title string  // title of movie
    year int     // movie release year
    genre string  // pipe-separated list of genres}
}

func createQuery(q interface{}) {
    if reflect.ValueOf(q).Kind() == reflect.Struct {
        v := reflect.ValueOf(q)
        fmt.Println("Number of fields", v.NumField())
        for i := 0; i < v.NumField(); i++ {
            fmt.Printf("Field:%d type:%v value:%v\n", i, v.Field(i).Type(), v.Field(i))
        }
    }
}

func main() {
    joker := movie {movieId: 193612, title:"Joker",
                    year: 2019, genre: "action|thriller"}
    createQuery(joker)
}
```

- Try this example:

https://go.dev/play/p/EIWbrp_kSXb

- Extend this example with links struct

```
type links struct {
    movieId int    // represents the movie id
    imdbId  int    // IMDB code
    tmdbId  int    // TMDB code
}
```

Go Reflection -

reflect.Int, reflect.String

```
package main
```

```
import (  
    "fmt"  
    "reflect"  
)
```

```
func main() {  
    movieId := 193612  
    id := reflect.ValueOf(movieId).Type() -> Int()  
    fmt.Printf("type: %T value: %v\n", id, id)  
    movieTitle := "Joker"  
    title := reflect.ValueOf(movieTitle).String()  
    fmt.Printf("type: %T value: %v\n", title, title)  
}
```

- Try this example:

https://play.golang.org/p/_mOVAjvkkuQ

Go Reflection - Exercise

- Write a program that uses the reflect API to manipulate (query, insert into) the MovieLens database
- Modify your go code from the SQL precept
- Update your precept repo:

```
cd <Precepts repo>
```

```
git pull                # update with precept7
```

```
cd precept7
```

- Modify movies.go to use reflection to form queries, for example:
 - One function for inserting movies or ratings into the appropriate table
 - One function for retrieving a movie or rating based on movied