RESTful Web Service in Golang

Joseph Devita & Daoun Oh

Usage

Webservice:

```
	GOPATH$ go get webservice
	GOPATH$ go install webservice
	GOPATH$ webservice
```

Client:

```
	GOPATH/src/client$ go run client.go [-url=<url>] [-data=<student object>] [-year=<year>] [-
method=<method>]
```

File Structure:

```
GOPATH
	/go
		/src
			/client
				Client.go
			/webserivce
				Server.go
				Student.go
				Handlers.go
		/bin
		/pkg
```

Methods

<u>ADD STUDENTS</u> – Adds a new student to the database with specified student information

Command-line: `go run client.go -url="http://localhost:1234/Student" -method=create -data='{student}'`
Expected output: {MongoID NetId Name Major Year Grade Rating}

<u>LIST STUDENT</u> – gives information about a specified student (specified by student name)

Command-line: `go run client.go -url="http://localhost:1234/getstudent?<field>=<value>" -method=list`
Expected output: {MongoID NetId Name Major Year Grade Rating}

<u>DELETE STUDENT</u> – deletes students by a specified year

Command-line: `go run client.go -url="http://localhost:1234/Student" -method=remove -year=<year>`
Expected output: #of students removed

<u>LIST ALL STUDENTS</u> – gives a list of all students in the database
Command-line: `go run client.go -url="http://localhost:1234/Student/listall" -method=list`
Expected output: list of all students in database in this format
{MongoID NetId Name Major Year Grade Rating}

<u>UPDATE</u> – updates students' ratings by calculating the average of all of the students' grades and giving them a rating based off from the calculated average
Command-line: `go run client.go -url="http://localhost:1234/Student" -method=update`
Expected Output: Class average is: average value

Test Cases

```
INPUT: $ go run client.go -url="http://localhost:1234/getstudent?name=mike" -method=list

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ea771492e891d86c14416") Mike Computer Science 2015 69 B}


INPUT: $ go run client.go -url="http://localhost:1234/Student/listall" -method=list

EXPECTED OUTPUT: list of all students in database in this format
                    {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT:
{ObjectIdHex("563ea771492e891d86c14416") 147001238 Mike Computer Science 2015 69 B}
{ObjectIdHex("563ea818492e891deae6da63") 147001239 Mike Computer Science 2015 70 B}
{ObjectIdHex("563ea81e492e891deae6da64") 147001231 Mike Computer Science 2015 80 B}
{ObjectIdHex("563ea823492e891deae6da65") 147001232 Mike Computer Science 2015 50 C}
{ObjectIdHex("563ead79492e891f2baf10c5") 147001233 Mike Computer Science 2015 90 A}
{ObjectIdHex("563ead7a492e891f2baf10c6") 147001234 Mike Computer Science 2015 90 A}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=remove -year=2015

EXPECTED OUTPUT: #of students removed
ACTUAL OUTPUT: 6 student(s) removed


INPUT: $ go run client.go -url="http://localhost:1234/Student/listall" -method=list

EXPECTED OUTPUT: list of all students in database in this format
                    {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT:


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
90,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb191a3a6902b34fde10") 147001238 Mike Computer Science 2015 90 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
80,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb221a3a6902b34fde11") 147001238 Mike Computer Science 2015 80 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
86,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb2c1a3a6902b34fde12") 147001238 Mike Computer Science 2015 86 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
```

```
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
66,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb311a3a6902b34fde13") 147001238 Mike Computer Science 2015 66 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
70,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb3a1a3a6902b34fde14") 147001238 Mike Computer Science 2015 70 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=create -
data='{"NetID":"147001234","Name":"Mike","Major":"Computer Science","Year":2015,"Grade":
50,"Rating":"D"}'

EXPECTED OUTPUT: {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT: {ObjectIdHex("563ecb3f1a3a6902b34fde15") 147001238 Mike Computer Science 2015 50 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student/listall" -method=list

EXPECTED OUTPUT: list of all students in database in this format
                    {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT:
{ObjectIdHex("563ecb191a3a6902b34fde10") 147001234 Mike Computer Science 2015 90 D}
{ObjectIdHex("563ecb221a3a6902b34fde11") 147001235 Mike Computer Science 2015 80 D}
{ObjectIdHex("563ecb2c1a3a6902b34fde12") 147001236 Mike Computer Science 2015 86 D}
{ObjectIdHex("563ecb311a3a6902b34fde13") 147001237 Mike Computer Science 2015 66 D}
{ObjectIdHex("563ecb3a1a3a6902b34fde14") 147001238 Mike Computer Science 2015 70 D}
{ObjectIdHex("563ecb3f1a3a6902b34fde15") 147001239 Mike Computer Science 2015 50 D}


INPUT: $ go run client.go -url="http://localhost:1234/Student" -method=update

EXPECTED OUTPUT: Class average is: average value
ACTUAL OUTPUT: Class average is:  73


INPUT: $ go run client.go -url="http://localhost:1234/Student/listall" -method=list

EXPECTED OUTPUT: list of all students in database in this format
                    {NetId Name Major Year Grade Rating}
ACTUAL OUTPUT:
{ObjectIdHex("563ecb191a3a6902b34fde10") 147001234 Mike Computer Science 2015 90 A}
{ObjectIdHex("563ecb221a3a6902b34fde11") 147001235 Mike Computer Science 2015 80 B}
{ObjectIdHex("563ecb2c1a3a6902b34fde12") 147001236 Mike Computer Science 2015 86 A}
{ObjectIdHex("563ecb311a3a6902b34fde13") 147001237 Mike Computer Science 2015 66 B}
{ObjectIdHex("563ecb3a1a3a6902b34fde14") 147001238 Mike Computer Science 2015 70 B}
{ObjectIdHex("563ecb3f1a3a6902b34fde15") 147001239 Mike Computer Science 2015 50 D}
```

Development Peculiarities

Implementing the Student struct supplied on the assignment page caused issues when attempting to store the data in Mongo. We lost the NetID field when attempting to store the struct in a “raw” form and marshaling it via the BSON library. A new field for the mongoID was thus created, and the create handler checks for a distinct NetID before adding an object. The documentation for the GOLANG mongo API wasn’t as thorough as standard GOLANG libraries. Advanced query logic, specifically mongo operators wasn’t present in a substantial way and outside sources (stack overflow) were relied on entirely for said areas.

Note:The submission under Joseph Devita’s Sakai should not be graded over this version.

High Level Data Analysis

A session is created with a mongoldb server that is used for at my tech-startup:

```
mgo.Dial(“mongodb://dsproject:password@dogen.mongohq.com:10052/bow-ties-are-hard-to-tie")
```

If it is required that you can see the contents of the database, go to compose.io and use login:

jodvita@gmail.com
Supersecure1

Routes and associated session logic are all concisely declared in Server.go:

```
       r.GET("/getstudent", uc.GetStudent)

       r.GET("/Student/listall", uc.ListStudents)

       r.POST("/Student", uc.CreateStudent)

       r.DELETE("/Student", uc.RemoveStudent)

       r.GET("/Student", uc.UpdateStudents)
```

These routes are associated with methods in Handler.go. Each method described in the Method’s section is Implemented in it’s own function. Student.go houses the structure of a student object in JSON form; it also contains the appropriate mongoDB overrides:

```
type (
    Student struct{
        ID      bson.ObjectId   `json:"id" bson:"_id"`
        NetID   string          `json:"netid" bson:"netid"`
        Name    string          `json:"name" bson:"name"`
        Major   string          `json:"major" bson:"major"`
        Year    int             `json:"year" bson:"year"`
        Grade   int             `json:"grade" bson:"grade"`
        Rating  string          `json:"rating" bson:"rating"`
    }
)
```

RESTful calls are executed via calling client.go with command line arguments. A request corresponding to the arguments is created, and the response of the service(if any) is outputted via standard out of the client. All actions are atomic, as a result after client.go outputs the results of the operation the client terminates.