Module 1, Lesson 2: MongoDB Ruby Driver CRUD

The overall goal of the assignment is to:

- implement various document access methods using the MongoDB Ruby Driver

The functional goal of the assignment is to:

- implement various document access methods for Race Results

Note that this assignment was written so that you can implement it in parts after each lecture. If you are performing the assignment in between lectures, stop at the next lecture boundary in the technical requirements section and resume once you have completed the lecture. You are free to experiment with other forms of the queries presented, but the grading will only be targeted at the specific requirements listed.

Getting Started

1. Start your MongoDB server using `mongod`
2. Download and extract the starter set of files. The root directory of this starter set will be referred to as the root directory of your solution.

```
--- student-start
    |-- assignment.rb
    |-- race_results.json
    |-- .rspec (important hidden file)
    `-- spec
        |-- lecture1_spec.rb
        |-- lecture2_spec.rb
        |-- lecture3_spec.rb
        |-- lecture4_spec.rb
        |-- lecture5_spec.rb
        `-- spec_helper.rb
```

   - assignment.rb - your solution must be placed within this file
   - spec - this directory contains tests to verify your solution. You should not modify anything in this directory

3. Install the following gems. You may already have them installed.

```
$ gem install rspec
$ gem install rspec-its
$ gem install mongo -v 2.1.2
```

4. Run the rspec command from the project root directory (i.e., `student-start` directory) to execute the unit tests within the spec directory. This should result in several failures until you complete your solution in `assignment.rb`.

```
$ rspec

(N) examples, (N) failures, (N) pending
...
```

5. Implement the Ruby technical requirements in `assignment.rb` within the provided class `Solution`. Helper methods have been provided to get a connection to Mongo and to set the database and collection names. You can override these values using environment variables if you are not using the defaults. Since you are not turning in this assignment – you may also simply edit the defaults used within the source file. However, the defaults used are as follows:

- MONGO_URL='mongodb://localhost:27017'
- MONGO_DATABASE='test'
- RACE_COLLECTION='race1'

7. You can load the `assignment.rb` script into the `irb` shell and take advantage of the helper methods of the `Solution` class to try out commands in the shell before adding them to the assignment solution.

```
$ irb
> require './assignment.rb'
 => true
> racers=Solution.collection
  => #<Mongo::Collection:0x8262360 namespace=test.race1>
> racers.find.count
  => 1000
```

Implement all methods relative to the `@coll` instance variable setup to reference the collection.

Technical Requirements

As you implement each requirement, the following is provided to help familiarize you with the schema associated with a race document:

```
{"_id"=>BSON::ObjectId('564c01c886c12c3d3d0003ca'),
 "number"=>970,
 "first_name"=>"LONNIE",
 "last_name"=>"FITZGERALD",
 "gender"=>"F",
 "group"=>"14 and under",
 "secs"=>2258}
```

Lecture 1: Create In this section we will use `delete_many`, `insert_many`, and `insert_one` to implement and test different create metthods.

1. Implement an instance method called `clear_collection` that:
   - accepts no arguments
   - deletes all documents from the collection (hint: `delete_many`)
   - returns the Mongo result object for the command

   You can demonstrate your method using the `irb` shell.

```
> Solution.collection.insert_one({})
 => #<Mongo::Operation::Result:22500120 documents=[{"ok"=>1, "n"=>1}]>
> s=Solution.new
> r=s.clear_collection
 => #<Mongo::Operation::Result:22249400 documents=[{"ok"=>1, "n"=>1}]>
> r.ok?
 => true
> r.deleted_count
 => 1
```

```
$rspec spec/lecture1_spec.rb -e rq01
```

2. Implement an instance method called `load_collection` that:
   - accepts a file_path argument to a file of JSON data containing race results
   - reads the JSON contents of the file into an array of hashes. You may use the `load_hash()` method provided in the bootstrap `assignment.rb` for this.

- inserts each of the hash elements of the array into the database collection (hint:insert_many)
- returns the Mongo result object for the command

You can demonstrate your method using the `irb` shell.

```
> r=s.load_collection('./race_results.json')
> r.inserted_count
 => 1000
> r.inserted_ids.slice(0,2)
 => [BSON::ObjectId('5663a565e301d093ac0003eb'), BSON::ObjectId('5663a565e301d093ac0003ec')]
```

```
$rspec spec/lecture1_spec.rb -e rq02
```

3. Implement an instance method called `insert` that:

   - accepts a hash for race result data
   - inserts this one race result into the collection (hint:insert_one)
   - returns the Mongo result object for the command

   You can demonstrate your method using the `irb` shell.

```
> r=s.insert({:foo=>"bar"})
 => #<Mongo::Operation::Result:18276240 documents=[{"ok"=>1, "n"=>1}]>
> r.inserted_ids
 => [BSON::ObjectId('5663a66fe301d093ac0007d3')]
2.2.2 :052 > r.returned_count
 => 1
```

```
$rspec spec/lecture1_spec.rb -e rq03
```

Lecture 2: Find By Prototype In this section we will locate documents in a collection based on the AND'ing of extact-matching parameters acting as `prototype` documents. We will also begin to shape the returned document(s) with the use of `projection`.

1. Implement an instance method called `all` that:

   - accepts an optional hash prototype
   - finds all documents that match all parameters in the hash (or all documents if empty hash). In this case, the caller is required to form the hash for the query that matches the field names. (hint: find)
   - returns the Mongo result object for the command with all fields of the document included

   You can demonstrate your method using the `irb` shell.

```
> s.all(:first_name=>"MARY").first
 => {"_id"=>BSON::ObjectId('5663a755e301d099140003d0'), "number"=>975, "first_name"=>"MARY",
        "last_name"=>"RODGERS", "gender"=>"M", "group"=>"30 to 39", "secs"=>3025}
```

```
$rspec spec/lecture2_spec.rb -e rq01
```

2. Implement an instance method called `find_by_name` that:

   - accepts a first and last name
   - finds all documents that match the first and last name provided. In this case, you must actually form the hash for the query. (hint: find)
   - forms a projection that returns only the `first_name`, `last_name`, and `number` properties (hint: projection
   - returns the Mongo result object for the command

   You can demonstrate your method using the `irb` shell.

```
> s.find_by_name("MARY", "RODGERS").first
 => {"number"=>975, "first_name"=>"MARY", "last_name"=>"RODGERS"}
```

```
$rspec spec/lecture2_spec.rb -e rq02
```

Lecture 3: Paging In this section we will get some practice bounding method results based on sorting and paging commands.

1. Implement an instance method called `find_group_results` that

   - accepts a `group` name, `offset` value, and `limit` value
   - finds only race results for the specified `group`
   - forms a projection that eliminates the `group` and `_id` fields from the results (hint: `projection`)
   - sorts the results by time (`secs`), accending (hint: `sort`)
   - skips `offset` documents in the ordered result (hint: `skip`)
   - limits the results to only `limit` documents (hint: `limit`)
   - returns the Mongo result object for the command

   You can demonstrate your method using the `irb` shell.

```
> r=s.find_group_results("30 to 39", 1, 2)
> r.to_a.count
 => 2
> r.to_a
 => [{"number"=>248, "first_name"=>"MARTIN", "last_name"=>"RAMOS", "gender"=>"M", "secs"=>1263},
     {"number"=>320, "first_name"=>"TWILA", "last_name"=>"TUCKER", "gender"=>"F", "secs"=>1270}]
> r.selector
 => {:group=>"30 to 39"}
> r.count
 => 129
```

```
$rspec spec/lecture3_spec.rb
```

Lecture 4: Find By Criteria In this section we will find documents based on a comparison that does not involve equality.

Hint: Review the Mongo Query and Projection Operators

1. Implement an instance method called `find_between` that

   - accepts a `min` and `max` value
   - finds all race results with a time (`secs`) that is between `min` and `max` (exclusive).

   You can demonstrate your method using the `irb` shell.

```
> r=s.find_between(1600,1615)
> r.to_a.count
 => 2
> r.to_a
 => [{..."first_name"=>"MEGAN", "last_name"=>"ANDERSON"... "secs"=>1605},
     {..."first_name"=>"KELI", "last_name"=>"SIMPSON"... "secs"=>1606}]
```

```
$rspec spec/lecture4_spec.rb -e rq01
```

2. Implement an instance method called `find_by_letter` that

   - accepts a `letter`, `offset`, and `limit`
   - finds all race results with the last_name that starts with the `letter` provided using a regular expression. You only need to treat `letter` as a string and do not have to enforce as a character. However, you should convert this value to upper case. The following REGEX `"^S.+"` will locate all names starting with the letter `S`.
   - orders the results by `last_name`, ascending
   - skips the first `offset` documents
   - limits results to `limit` documents
   - returns the Mongo result object for the command

You can demonstrate your method using the `irb` shell.

```
r=s.find_by_letter("sal",0,1).first
 => {..."first_name"=>"LOURIE", "last_name"=>"SALAZAR"...}
```

```
$rspec spec/lecture4_spec.rb -e rq02
```

Lecture 5: Update In this section we will modify the database.

1. Implement an instance method called `update_racer` that
   - accepts a hash of racer properties
   - finds the racer associated with the `_id` property in the input hash
   - replaces all existing fields for the racer with what is provided (hint: `replace_one`)
   - returns the Mongo result object for the command

   You can demonstrate your method using the `irb` shell.

   ```
   > r1=s.all.first
    => {"_id"=>..., "number"=>0, "first_name"=>"SHAUN", "last_name"=>"JOHNSON" ...
   > r1[:first_name]="foo"; r1[:last_name]="bar"
   > r1
    => {"_id"=>..., "number"=>0, "first_name"=>"foo", "last_name"=>"bar" ...
   > r=s.update_racer(r1)
    => #<Mongo::Operation::Result:28472340 documents=[{"ok"=>1, "nModified"=>1, "n"=>1}]>
   > r.modified_count
    => 1
   ```

   ```
   $rspec spec/lecture5_spec.rb -e rq01
   ```

2. Implement an instance method called `add_time` that
   - accepts the racer number and an amount of time in seconds
   - finds the racer's document and increments the time in the database without retrieving the actual document. (hint: `:$inc`)

   You can demonstrate your method using the `irb` shell.

   ```
   r1=s.all.to_a.sample
    => {"_id"=>BSON::ObjectId('5663a754e301d09914000054'), "number"=>83, ... "secs"=>3146}
   > r=s.add_time(r1[:number], 1000)
    => #<Mongo::Operation::Result:23028380 documents=[{"ok"=>1, "nModified"=>1, "n"=>1}]>
   > s.all(:number=>r1[:number]).first[:secs]
    => 4146
   ```

   ```
   $rspec spec/lecture5_spec.rb -e rq02
   ```

Self Grading/Feedback

Unit tests have been provided in the bootstrap files that can be used to evaluate your solution. They must be run from the same directory as your solution.

```
$ rspec
........

(N) examples, 0 failures
```

Submission

There is no submission required for this assignment but the skills learned will be part of a follow-on assignment so please complete this to the requirements of the unit test.

Last Updated: 2015-01-19