[GitHub Database Usage]

[Board and Card Game App Team]

[Wyatt Derk]

[2/15/19]

[How can GitHub be used as a database?]

With the use of GitHub as our team's version control, it would be convenient, if it could be used as the app's database as well. The question is whether that is possible. With githubDb, this is possible.

To begin, the developer will need a personal token. This can be acquired by going to this link: https://github.com/settings/tokens/new. Next the module needs to be installed locally. Below are the steps that the developer will need to undertake.

```
$ npm install github-db
```

```
/**

* We are going to authenticate with Github and

* specify our repo name and file we just created.

*/

var options = {

  host: 'private-github-api.com', // <--
 Private github api url. If not passed, defaults to 'api.github.
com'

  pathPrefix: 'prefix-for-enterprise-instance', // <--
 Private github api url prefix. If not passed, defaults to null.

  protocol: 'https', // <--
 http protocol 'https' or 'http'. If not passed, defaults to 'ht
tps'

  user: 'github-username', // <-- Your Github username

  repo: 'github-repo', // <-- Your repository to be used a db
```

```
  remoteFilename: 'filename-with-extension-json' // <-
 File with extension .json

};



// Require GithubDB

var GithubDB = require('..').default;

// Initialize it with the options from above.

var githubDB = new GithubDB(options);



// Authenticate Github DB -
> grab a token from here https://github.com/settings/tokens

githubDB.auth(personalAccessToken);



// Connect to repository

githubDB.connectToRepo();



// You are now authenticated with Github and you are ready to us
e it as your database.

githubDB.save({"message": "wooohoo"});
```

Here is a more detailed explanation on what is going on in this code:

First, the developer authenticates GitHub:

```
githubDB.auth(personalAccessToken)
```

Next, the personal access token is set as an environmental variable. Users can request tokens from GitHub [https://github.com/settings/tokens].

```
var personalAccessToken = process.env.TOKEN; // Set the variable
 here
```

Then, you start the server with the token.

```
$ TOKEN=xxxx node app.js
```

Then, you connect to the GitHub database (after being authenticated).

```
githubDB.connectToRepo();
```

When setting the file for database usage on GitHub, be sure to add a valid JSON array. If you don't , githubDB will return an empty array. It may also give you this error:

```
undefined:0



^

SyntaxError: Unexpected end of input
```

Writing and Saving to Collection:

Here a developer is able to save objects.

```
githubDB.save(users).then((res) => {

    // The result from the same

    console.log(res);

});
```

A method for saving multiple objects at once is as follows:

```
var article1 = {

    title : 'githubDB rocks',
```

```
    published : 'today',
    rating : '5 stars'
}
var article2 = {
    title : 'githubDB rocks',
    published : 'yesterday',
    rating : '5 stars'
}
var article3 = {
    title : 'githubDB rocks',
    published : 'today',
    rating : '4 stars'
}
githubDB.save([article1, article2, article3]).then((res) => {
    // The result from the same
    console.log(res);
});
```

Which would return the following:

```
[ { title: 'githubDB rocks',
    published: 'today',
    rating: '4 stars',
```

```
    _id: 'b1cdbb3525b84e8c822fc78896d0ca7b' },
  { title: 'githubDB rocks',
    published: 'yesterday',
    rating: '5 stars',
    _id: '42997c62e1714e9f9d88bf3b87901f3b' },
  { title: 'githubDB rocks',
    published: 'today',
    rating: '5 stars',
    _id: '4ca1c1597ddc4020bc41b4418e7a568e' } ]
```

Reading from the Collection:

These are the 3 methods for reading the JSON collection:

- db.collectionName.find(query)
- db.collectionName.findExact(query)
- db.collectionName.findOne(query)

Here we have an example of the first one. This would return all records.

```
githubDB.find().then((results) => {

    // Results here

    console.log(results);

});
```

With its return here:

```
[{

    title: 'githubDB rocks',
```

```
    published: 'today',

    rating: '5 stars',

    _id: '0f6047c6c69149f0be0c8f5943be91be'

}]
```

In addition, a developer can query with criteria. The query can even contain multiple criteria, as seen below.

```
githubDB.find({rating : "5 stars", published: "yesterday"}).then
((results) => {

    console.log(results);

});
```

This would return all articles with a rating of 5, that were published yesterday.

Next is githubDB.findExact(query). Here is an example of it in use:

```
githubDB.findExact({

        title: 'githubDB rocks',

        rating: '4 stars'

    }).then((results)=> {

    console.log(results);

});
```

This method will return an exact match of the criteria search for. Here is the its results:

```
[{

    title: 'githubDB rocks',

    published: 'today',
```

```
    rating: '4 stars',

    _id: 'b1cdbb3525b84e8c822fc78896d0ca7b'

}]
```

githubDB.findOne(query) can be used to yield the first article in the collection, should nothing pass the query.

```
githubDB.findOne({_id: '0f6047c6c69149f0be0c8f5943be91be'}).then
((results)=> {

    console.lo
```

Updating the Collection:

Here, a developer can update items in a collection.

```
githubDB.update(query, data, options).then((updated)=> {

    console.log(updated);  // { updated: 1, inserted: 0 }

});;
```

A developer can also update multiple objects at a time:

```
options = {

    multi: false, // update multiple - default false

    upsert: false // if object is not found, add it (update-
insert) - default false

}
```

Here is an example of the function at work.

```
var query = {

  title : 'githubDB rocks'
```

```
};


var dataToBeUpdate = {

  title : 'githubDB rocks again!',

};


var options = {

   multi: false,

   upsert: false

};


var updated = githubDB.update(query, dataToBeUpdate, options).th
en((results) => {

  console.log(updated); // { updated: 1, inserted: 0 }

});
```

Removing a Collection:

Here a developer can remove an entire collection or the objects that pass through a query.

```
githubDB.removeAll();

githubDB.remove(query, multi);
```

When removing by query, the all matched objects will be deleted, because multi will be set to true. Set multi to false in order to only delete the first one.

```
githubDB.remove({rating : "5 stars"});

githubDB.remove({rating : "5 stars"}, true); // remove all match
ed. Default - multi = true

githubDB.remove({rating : "5 stars"}, false); // remove only the
 first match
```

[https://www.npmjs.com/package/github-db] /*This was the site used to create the document.*/