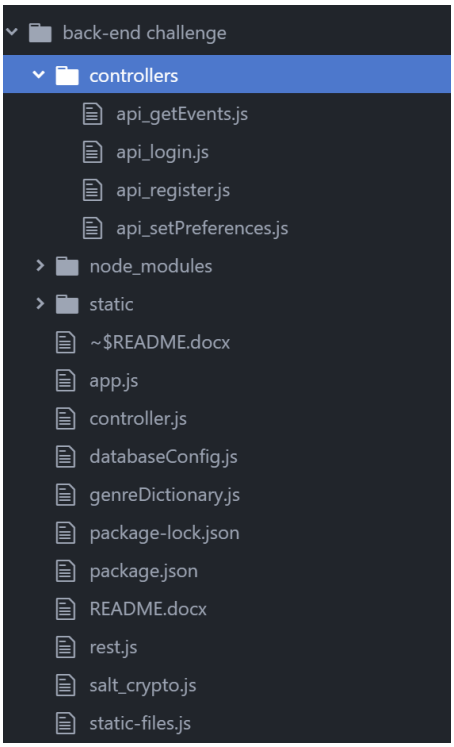
Here is the introduction of different files' utilities, how to use this program and displays of test case.

Introduction

This program uses mysql as database to store user information. There are only one table there, containing attributes (uid, email, password, name, birthdate, classificationName, genreId).

uid is the primary key for the table, which is automatically increased. Password will be encrypted by md5 so that we can prevent malicious revision on the database through client. Also we use mysql.escape to prevent database injection. ClassificationName is the category of events and genreId is the id of genre.

In this program, we use Koa as framework of the back-end. The directory of files are:



app.js: the main function to start the server.

controller.js: the file to process each .js file in /controllers.

databaseConfig.js: the file to record the database configuration information.

genreDirectory.js: the file to record the genre name and its relative genreId

package.json: program description file.

rest.js: the file to set up rules for API's error handling and response format.

salt_crypto.js: the file to define the function of encryption.

static-files.js: the file to handle the static file. I wrote this file to handle the test.html in ./static/ to realize a simple ajax to test APIs.

./controllers: contain APIs for each functions.

api_register.js: This file is the API for register.We assume the data received from client is:

```
data: {
        email,
        password,
        name,
        birthdate,
        classificationName,
        genre
}
```

the data send back to client is {isSuccess: true/false}, which indicates whether the register is successful or not.

api_login.js: This file is the API for login. We assume the data received from client is:

```
data: {
        email,
        password
}
```

The data send back to client is {isSuccess: true/false, uid}, which indicates whether the register is successful or not, and the uid of current user. The uid can be used in API getEvents and API setPreferences.

As for authentication checks for each subsequent API(/getEvents and /setPreferences), we can write the email and password in URL. Each time we run subsequent API, we can review information of the current user to gain authentication. However, to be simplified, here I just send uid of the current user back to client and each time when subsequent API calls, the client should send this uid back to server.

api_getEvents.js: This file is the API for getEvents. We assume the data received from client is:

```
data: {
        uid
}
```

The data send back to client is an object including information for events

api_setPreferences.js: This file is the API for setPreferences. We assume the data received from client is:

```
data: {
```

uid,

classificationName,

genre}

The data send back to client is {isSuccess: true/false}, which indicates whether the register is successful or not

Instruction

To run this program, we firstly have to run:

npm install

to install necessary modules like koa, mysql, etc.

Then run

node app.js

to start the server.

To test the APIs, I write a simple .html file to implement ajax to test the result for different APIs.

Test Case

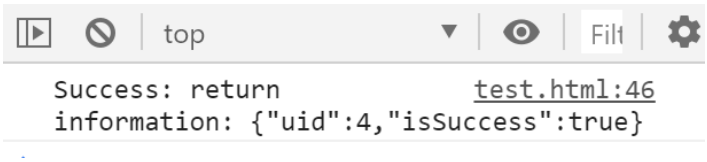
1. Register:

```
data: {
  email: "test01@example.com",
  password: "111111",
  name: "Mr test01",
  birthdate: "1995-01-01",
  classificationName: "Music",
  genre: "R&B"
},
```

Result in database:

2. Login:

```
data: {
  email: "test01@example.com",
  password: "111111"
},
```



3. getEvents:

```
data: {
  uid: 4
},
```

4. setPreferences:

```
data: {
  uid: 4,
  classificationName: "Film",
  genre: "Foreign"
},
```

	uid	email	password	name	birthdate	classificationName	genreId
Edit Copy Delete	4	test01@example.com	abb03c22b8be60fd06d5d29a2f7085e7	Mr test01	1995-01-01 00:00:00	Film	KnvZfZ7vAk1