Zeus Guide Book

Aim: this guide book is for user to use the CUI program that I have built

Usage:

1. Command options:

```
options:
  -o, --operation    operation on database (string)
  -t, --table        table name (string)
  -i, --id           id of entity (string [=])
  -c, --content      content of a entity (string [=])
      --help         print the help message
```

2. Explaination for command options

(1). –o, --operation, this option is for user to specify the operation they want to have on database. I provide four kind of operations to users, that is CRUD. For create, update, read and delete. Their relation is below

-o delete **delete operation on table**
-o create **create operation on table**
-o update **update operation on table**
-o read **read operation on table**

(2).For every operation, I have different function to call

-o delete

bool DeleteById(const string& table, const string& id)

input: table: the table name, eg region
id : the id of entity, eg 12

output: bool, indicate if this operation is successful

side effect: delete the entity

-o create

bool WriteToTable(const string& table, const string& content)

input: table, the table name, eg region
content, the content want to add, eg. 12,China,123

output: bool,indicate if this operation is successful

side effect: create a new entity

-o read

vector<string> FindById(const string& table, const string& id)

Input: table, the table name, eg. Region
Id, the entity id, eg. 12

Output: vector of string to indicate the entity' s content

-o update

bool UpdateById(const string& table, const string& id, const string& content)

input: table, table nam, eg. Region

id, the entity id, eg. 12

content, the content to updata. eg 12,Chinq,123

output: bool to indicate if this operation is successful

side effect: update a entity

(3).Explanation for table

Because this version is still not complete, so we can not have operations on every Table of hotel, we can just operate on these below tables

1. staff

2. customer

3. hotel

4. room

5. region

The table can be a key to get the table file path

Operation on table

-t table_name

Table_name is one of **{ "customer" ," staff" ," hotel" ," room" ," region" }**

(4).Explanation for id

-i/--id id

Id to indicate the entity

It is based on table

(5).Expalation for content

-c/--content content

Indicate the content will be added or update

3. How to realize

(1) Read operation

Class: CSVParse

Function: vector<string> find(const string& pattern, size_t pos)

Explanation: it is just like search, we use id to indicate a entity, and we search the table to find the responsive one

(2) Delete operator

Class: CSVParser,CSVWriter,ifstream,ofstream

Function: result_iterator CSVParse::begin()

Void CSVWriter::write(const string& content)

Bool SwapFiles(const string& inFileName, const string& outFileName);

Explanation: This is not as easy as read function, because the way we store data is not Database,it is a file, so we can not directly to delete it. We should firstly Migrate the file to a tmp file except the deleted one, and then migrate

Back.

(3) Create operation

Class: CSVWriter

Function: void CSVWriter::write(const string& content)

Explanation: It is easy, just add a new line to the end of original file

(4) Update operation

Class: CSVParser, CSVWriter ,ifstream, ofstream

Function: result_iterator CSVParse::begin()

Void CSVWriter::write(const string& content)

Bool SwapFiles(const string& inFileName, const string& outFileName);

Explanation: This is not as easy as read function, because the way we store data is not Database,it is a file, so we can not directly to update it. We should firstly Migrate the file to a tmp file except the deleted one, and then migrate Back.