VA Online Memorial - Data scraper improvements

Revision History

Author	Revision Number	Date
TCCODER	1.0	Feb 09, 2018

Deployment Instructions

Deployment Instructions

1. Deployment Dependencies

Before performing a deployment, it is assumed that the following have been set up:

- NodeJs 8+
- Postgresql Database
- Libpq (pg_config)

2. Organization of Submission

- *va-backend/ – source of the submission*
- *docs/ - the deployment guide*

3. Deployment Instructions

Go to the *va-backend/* folder and follow the instructions of the README.md file to install all packages.

To install libpq do one of the following:

On OS X: brew install postgres
On Ubuntu/Debian: apt-get install libpq-dev g++ make
On RHEL/CentOS: yum install postgresql-devel

Change the database configuration in the *packages/va-data-scraper* and *packages/va-models* packages to match your Postgresql database configuration.

Go to the *va-backend/packages/va-data-scraper* and follow the instructions of the README.md file information on how to run the package.

Don't run "npm install" inside the *va-backend/packages/va-data-scraper* folder.

4. Verification

Go to the folder *va-backend/packages/va-data-scraper*.

Run the command:

```
$ npm run download-data
```

Wait until it downloads 20 files and hit Ctrl+C (if you want to download all files go ahead but it's not necessary).

As described in https://apps.topcoder.com/forums/?module=Thread&threadID=912402, an option to ignore CSV error can be made.

For that set the environment variable **OPTION_IGNORE_BAD_CSV_LINE** to “true” and run the script (check *va-backend/packages/va-data-scraper/services/data.js* file for **OPTION_IGNORE_BAD_CSV_LINE** usage).

To start the data scraper script run:

```
$ export OPTION_IGNORE_BAD_CSV_LINE=true
$ npm run import-data
```

You will see something like this:

```
> node --expose-gc --max_old_space_size=4096 scripts/import-data.js

[2018-02-08T11:15:21.140Z][INFO] Will ignore bad csv lines
[2018-02-08T11:15:21.143Z][INFO] connecting to database: postgres://postgres:topcoder@localhost:5432/test
[2018-02-08T11:15:21.330Z][INFO] Processing file downloads/ngl_alabama.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 387.8s
0.0s
[2018-02-08T11:21:54.236Z][INFO] Parsed file downloads/ngl_alabama.csv with 21872 lines. Inserted 20070 veterans
[2018-02-08T11:21:54.255Z][INFO] Processing file downloads/ngl_alaska.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 155.9s
0.0s
[2018-02-08T11:24:31.838Z][INFO] Parsed file downloads/ngl_alaska.csv with 8880 lines. Inserted 8411 veterans
[2018-02-08T11:24:31.936Z][INFO] Processing file downloads/ngl_arizona.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 2740.8s
0.0s
[2018-02-08T12:10:29.041Z][INFO] Parsed file downloads/ngl_arizona.csv with 96485 lines. Inserted 95108 veterans
```

While running the files you can check that the memory consumption stays below 150Mb (4Gb+ before).

To verify that transaction is working properly, move all files from the *downloads/* folder to a temporary folder, but keep file *ngl_california.csv*.

Drop the Postgresql database and create it again:

```
psql> DROP DATABASE test;
psql> CREATE DATABASE test;
```

Run the import the command again with **OPTION_IGNORE_BAD_CSV_LINE** set to “false”.

```
$ export OPTION_IGNORE_BAD_CSV_LINE=false
$ npm run import-data
```

It will crash while reading a CSV file.

```
> node --expose-gc --max_old_space_size=4096 scripts/import-data.js

[2018-02-08T12:19:18.491Z][INFO] connecting to database: postgres://postgres:topcoder@localhost:5432/test
[2018-02-08T12:19:19.446Z][INFO] Processing file downloads/ngl_california.csv
[█——————————————————————————————————————————————————————————————————————————] 1% 0.0s 0.0s
[2018-02-08T12:22:42.783Z][ERROR] Failed to recover CSV line
[2018-02-08T12:22:42.785Z][ERROR] Failed to read file. Stack: Error: Invalid closing quote at line 1; found "M"
instead of delimiter ","
[2018-02-08T12:22:42.787Z][INFO] Operation completed!
```

Check the database tables. No data must be present.

Run the import the command again. This time the command will not crash and some warnings or errors will be printed in the screen.

While running this file, *ngl_california.csv*, (the biggest one available) you can check that the memory consumption stays below 600Mb (4Gb+ before). This is due to the Sequelize library and not with the line by line read.

Last requirement verification:

Third, the scraper will ignore importing any row that has no information in these columns: first/last name, birth/burial date and cemetery name/city/address. That does make our data complete, but also skips a lot of records. We want you to analyze those skipped rows and propose a different strategy for importing records that would yield better results (you can propose more than one).

For that set the environment **OPTION_IMPORT_EXTRA_DATA** to true and run the script again (check *va-backend/packages/va-data-scraper/services/data.js* file for **OPTION_IMPORT_EXTRA_DATA** usage).

Drop the Postgresql database and create it again. Run the commands below:

```
$ export OPTION_IGNORE_BAD_CSV_LINE=true
$ export OPTION_IMPORT_EXTRA_DATA=true
$ npm run import-data
```

You should see something like this:

```
[2018-02-08T00:15:14.259Z][INFO] Will ignore bad csv lines
[2018-02-08T00:15:14.261Z][INFO] Will try to import extra data
[2018-02-08T00:15:14.262Z][INFO] connecting to database: postgres://postgres:topcoder@localhost:5432/test
[2018-02-08T00:15:14.425Z][INFO] Processing file downloads/ngl_alabama.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 375.6s
0.0s
[2018-02-08T00:21:34.670Z][INFO] Parsed file downloads/ngl_alabama.csv with 21872 lines. Inserted 21848 veterans
[2018-02-08T00:21:34.684Z][INFO] Processing file downloads/ngl_alaska.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 153.2s
0.0s
[2018-02-08T00:24:09.535Z][INFO] Parsed file downloads/ngl_alaska.csv with 8880 lines. Inserted 8869 veterans
[2018-02-08T00:24:09.629Z][INFO] Processing file downloads/ngl_arizona.csv
[████████████████████████████████████████████████████████████████████████████████] 100% 3953.9s
0.0s
[2018-02-08T01:30:19.496Z][INFO] Parsed file downloads/ngl_arizona.csv with 96485 lines. Inserted 96474 veterans
```

Comparing the execution with the previous one we have:

File	Without Extra Flag	With extra Flag	Increase
ngl_alabama.csv	20070 inserted	21848 inserted	8.8%
ngl_alaska.csv	8411 inserted	8869 inserted	5.4%
ngl_arizona.csv	95108 inserted	96474 inserted	1.4%

Checking the results, it can be verified an increase in the amount of imported data in all cases.

The following rules have been applied:

1- If column **relationship** is empty and names are equal to v_names we set relationship to **Veteran (Self).**
2- If column **relationship** is empty and last name is equal to v_last_name we set relationship to **Other Relative.**
3- If the **v_name** or v_last_name columns are empty and relationship is equal to **Veteran (Self),** we copy names to v_names.
4- If birth date or death date are null, we calculate the veteran id using a MD5 hash of the entire CSV line. This could produce a duplicate result, but it's very unlikely.

5. Resource Contact List

Name	Resource Email
TCCODER	Through TopCoder Member Contact