12/14/2017

A RESTful framework for running and evaluating code

A setup and installation guide

Ban, Christopher
SOUTHERN ADVENTIST UNIVERSITY

1 TABLE OF CONTENTS

ENVIRONMENT INSTALLATION AND SETUP

REQUIREMENTS

- Ubuntu 16.04.2 x64 Server
- Node (v8.9.1)
- Python (v3.5.2)
- g++-5 (v5.4.0)
- forever (optional)

2 INSTALLATION

(If fresh OS) Update distros/packages:

Run the following commands:

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
```

2.1 INSTALLING NODE:

Run the following commands:

```
$ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

2.2 INSTALLING PYTHON:

As of Ubuntu 16.04 LTS, Python 3 comes installed and is the default.

2.3 INSTALLING G++:

For C++ support, run the following commands:

```
$ sudo apt-get install g++-5
```

3 SETUP/CONFIGURATION

3.1 DOWNLOADING PROJECT FILES:

https://github.com/chrisban/cs-eval-fw

Run the following command in the appropriate directory: *(or unzip local archive of repository)*

```
$ git clone https://github.com/chrisban/cs-eval-fw.git
```

3.2 Installing package dependencies

(dependencies are specified in package.json)

cd into repository working directory and run the following command:

```
$ npm install
```

3.3 OPTIONAL PACKAGE INSTALLS:

Instead of starting via `$ node server.js:`

Forever: A simple CLI tool for starting a service and ensuring that the given script runs continuously in the background. See the documentation for details: https://www.npmjs.com/package/forever

cd into repo directory and run the following commands:

```
$ npm install forever

$ forever start server.js
```

3.4 SERVER CONFIGURATION:

Setting time zone (for optional date/time exam restrictions):

Run the following commands:

```
$ sudo timedatectl set-timezone <TIMEZONE>
```

Example: `$ sudo timedatectl set-timezone EST`

4 Data-file formatting

4.1 Format details

The data-files that define an activity follow the Javascript Object Notation (JSON) format.

For questions, the following structure is made up of eight fields. The "questionType" field can hold two values, "code" and "mchoice". This value defines whether the question is a coding question (where students are expected to write code), or a multiple choice question with randomized answers (where students are expected to answer questions; e.g., what result is produced by a given code snippet). Each field serves the same function for either type of question, excluding the "input" and "output" fields.

The "language" field defines the programming language used, "problem" defines the question, "skeleton" defines the initial skeleton code, "points" provides the point value, and "difficulty" defines a difficulty level which translates into minutes (value * 10 = suggested time) used to produce suggested completion times. The "skeleton" field also includes an optional usage which holds a "hidden skeleton", a replace token, and a "visible skeleton" for questions that may want to hide some details from the user, instead of just the standard skeleton. We have shown this optional format in line five of the example data structure below in section 4.3.

For coding questions, the "input" field utilizes an array that holds different test-cases. This input value can represent any input a keyboard can provide in a command-line setting. Each element in the "input" array is a different test case that will be run against student's submissions. The "output" field represents the expected output produced after running each test case using inputs (if any) from the "input" field.

Multiple choice questions on the other hand, use the "input" field differently. With this question type, the "problem" field describes the overall instructions and the "skeleton" field describes the corresponding code snippet, identical to coding questions (except it is non-editable). However, "input" utilizes a multi-dimensional array, with each internal array holding a sub-question and an array of options from which students can choose. The "output" field dictates the indices of the correct options.

The structure also contains a required "prop" field which can hold both required and optional information about the activity. The required fields "time" and "warn" are used to control how long an exam will last before auto submitting. The optional fields "closeDate" and "closeTime" denote when an exam will restrict access, "allowMultiple" defines whether students can take an exam multiple times, and "whitelist" and "access" control access to the exam. Note that the server's internal time-zone may not match your own, which can cause date specific properties to work unexpectedly.

4.2 Additional information:

- The 'problem' field can utilize html formatting/markup.
- If any double quotes are repeated, they must be escaped (ex: `"skeleton": " cout << \"i\"; "`)
- Generally, fields should not contain hard line breaks with the 'skeleton' field being the exception. In the skeleton field's case, the server can parse unformatted text wrapped in the following manner: '`_<<_ CONTENT _>>_`'
    - (Alternatively, explicitly write /n, /t, etc. instead of line breaks, as seen in the next section).

4.3 DATA-FILE EXAMPLE:

4.3.1 *Without* line breaks (pure JSON):

```
{
	"0": {
		"questionType": "code",
		"language": "c++",
		"problem": "Write a C++ function called print that accepts an integer and prints it out.",
		"skeleton": ["#include <iostream>\nusing namespace std;\n\n%READ_IN%\nint
		main() {\n string x = \"printed\";\n print(x);\n}\n", "%READ_IN%", "//Write your
		code here\nvoid print(string x){\n cout << x;\n}"],
		"input": ["1"],
		"output": ["1"],
		"points": ["15"],
		"difficulty": ".5"
	},
	“n”: {
		…
		…
		…
	},
	"prop": {
		"closeDate": "6-15-2018",
		"closeTime": "13:30",
		"allowMultiple": "false",
		"time": "15",
		"warn": ["10", "5"],
		"whitelist": ["0421291", "0421292"],
		"access": "true"
	}
}
```

4.3.2 *With* linebreaks (JSON-like):

```
{
	"0": {
		"questionType": "mchoice",
		"language": "c++",
		"problem": "What does the following program print when the user enters the indicated
		values?<br />(<i>NH means \"answer Not Here.\"</i>)",
		"skeleton": "_<<_#include <iostream>
		using namespace std;

		int main() {
			int i, j, k;
			cin >> i >> j >> k;
			// i, j, and k are ints
			if (i < j) {
				if (j < k)
					i = j;
				else
					j = k;
			} else {
				if (j > k)
					j = i;
				else
					i = k;
			}
			cout << \"i = \" << i << \" j = \" << j << \" k = \" << k << endl;
		} _>>_",
		…
	},
	…,
	…,
	…
}
```