Wilfrid Laurier University

CP493A Winter 2017

Setting Student Grades & Feedback Using The Brightspace API

Author:

Harold Hodgins

Supervisor:

David Brown

April 30, 2017

Last Updated January 16, 2019

1 Abstract

We created a simple program to help instructors and markers at Wilfrid Laurier University bulk upload student grades and feedback to My Learning Space using the Desire to Learn Brightspace API (application program interface). The program is course agnostic and should work with any course at Laurier which has grades on MLS or any university using a Brightspace instance.

The code is available at https://bitbucket.org/harohodg/brightspace-api-code. A tutorial for getting started with and using the BrightSpace API can be found at https://cocalc.com/projects/9501f241-b52e-43f8-9034-7292e8ee54ce/files/tutorial/BrightSpace_API.pdf

Contents

2 Background

Wilfrid Laurier University uses an online learning management system called My Learning Space (MLS) which is a re-branded version of a product called Brightspace from a company called Desire to Learn (D2L). It can be used to "add courses, content, or discussions with a single click. Run online, blended, and CBE[competency-based education] programs on the same platform. [And] deliver the right content at the right time with adaptive learning..."[1]

Many courses at Laurier use MLS to record student grades, assess students using quizzes, and accept student assignments. Some courses at Laurier also use MLS to provide sample assignment answers and feed back on student assignments. As an example CP164 - Data Structures I had over 200 students enrolled this semester and used MLS to accept students assignments for marking, provide students with sample assignment solutions, and to record student grades for labs, assignments and midterms.

Constructive student feedback is important for student comprehension of course material. In courses with labs this feedback is immediate. In contrast assignments submitted on MLS only garner feedback after the assignment is submitted. Although instructors can bulk upload student grades using a comma delimited file but any feedback has to be entered manually for every student individually through the grade item or dropbox interface.

This project aims to make it easy for those providing feedback on MLS to do so.

D2L provides API access to Brightspace instances with support for several major programming languages including Python, C#, Java SE, Javascript, and PHP [2]. Anything that can be done manually using a modern web browser can be done using the API without user intervention other then logging in initially [1].

[1] It is possible to do a headless login but it requires capturing and storing user credentials prior to usage.

3 Previous Work

3.1 Methods For Setting Feedback on MLS

3.1.1 Manually

Grades and feedback can be manually copied and pasted into MLS after it has been generated automatically by a computer or by manually by a marker. This is the current standard for hand marked and auto graded assignments in Computer Science courses at Laurier.

3.1.2 Grade All Interface

A naive method for automating setting student feedback on MLS is to use a program to click on the feedback icon for each student in the grade all interface, enter the feedback, and click Save. Then repeat with the next feedback icon. This requires loading the web page, finding the correct buttons and text boxes, and clicking on/filling them in accordingly. This is hard to do since the feedback text box is embedded in an iframe making it difficult to find and inject text into.

3.1.3 Dropbox Interface

The dropbox interface also has the ability to set grades and feedback. Any feedback set here forward propagates(as long as the out of values are set correctly) to the associated grade item. Feedback set in the grade item interface does not seem to back propagate.

The dropbox interface includes a Javascript function called feedback.setText(). This means it is easy to set feedback, then we can enter a grade, click on Update or Publish (depending if there was previously existing feedback), and then click on Next Student. This repeats until there isn't a Next Student button. Parsing the page we can get the students MLS internal ID # and use that to find their grade. We don't use the students name since parsing it is complicated by multitudinous variations. We don't use the Laurier ID # because the default format for downloaded student files from the dropbox includes the MLS ID # and not the Laurier ID #[2]. The primary down side to this method is that we need to hide the dropbox and the grade item if we aren't ready to release the grades immediately.

3.2 Dropbox Interface Implementation # 1

We started by using the Python PyQt framework to create a browser which would open the MLS homepage and wait till a page with a student dropbox item was opened. It would then attempt to find the student MLS ID #, fill in the correct boxes and click next until no students were left.

The primary problem we encountered was due to race conditions. If buttons were clicked too quickly the webpage and the program got out of sync and the program crashed ungracefully. The solution was to add timers and to check if an item existed before clicking.

The second problem we encountered was assuming the grade text box was always < **input id="z_bx"...**> in the page html but it isn't. This meant the program sometimes clicked on RECORD AUDIO, and then kept going without entering a grade. The solution was to search for a text box with a label immediately after which was "/#" eg /45.

This worked for most of the Fall 2016 semester but broke the first time we tried to use it this semester. It just sat there and did nothing. I suspect D2L changed the Dropbox interface in some subtle way.

[2]The default file name format can be changed by the server admins.

3.3 Improved Implementation

The temporary solution was to rewrite the browser interaction code from scratch using the Selenium WebDriver. By using a proper browser (Firefox on Ubuntu 12.04) it was much more tolerant to any changes made by D2L. We used the same overall algorithm to inject student grades and feedback one student at a time. The code took ~ 5 seconds per student to run because of the timing delays.

4 Current Work

4.1 Program Overview

This semester we restarted from scratch and created a program to enable markers and instructors to bulk upload students grades and feedback to MLS using the D2L Brightspace API. Using the API makes the program tolerant to any changes D2L makes to the MLS browser interface and also ensures the code is course agnostic and compatible with any course at Laurier with grade items on MLS or any at any university which uses Brightspace.

The program is written in Python 3.4 and uses a Flask micro server to interact with the user via any modern web browser. After the program is started the user begins by going to http://localhost:8080 which redirects to a MLS login page. After successfully logging into MLS the user is presented with a list of courses and associated grade items which they have access. Clicking on a grade item prompts them to upload a file of user feedback and grades which the program tries to parse and submit the contents to MLS. After uploading the grades and feedback the user is then shown any items that failed to upload and is presented with an option to logout, upload more grades, or check the grades that were just submitted. If the user tries to upload invalid grades (eg out of 20 but the grade item is out of 10) the user is given a link to update the grade item.

4.2 Program Development

The program was developed on a Lubuntu 14.04 virtual machine with Python 3.4 and the following libraries (many of which are part of Flask[3]Bottle and Beaker are only needed if

[3](

running the orignal D2L sample code.)).

- Beaker==1.8.1
- bottle==0.12.13
- click==6.7
- d2lvalence==1.2.2
- D2LValence-Util==0.1.16
- Flask==0.12.2
- future==0.16.0
- itsdangerous==0.24
- Jinja2==2.9.6
- MarkupSafe==1.0
- requests==2.13.0
- Werkzeug==0.12.2

Access to the API was provided by our local MLS support team. We gave them a trusted url (return route) and they gave us an application ID and application key. They also provided us with a course on the MLS test server and added several instructional assistants as students

4.3 Problems Encountered

4.3.1 Provided Example Code

We tested the sample code from https://github.com/Brightspace/sample-valence-general-python and tried to log in with the credentials given when the code was run and the website returned `Internal Error` The code itself was useful as a starting template.

4.3.2 Trusted URL

When initially testing the API we set the trusted url as `localhost:8080/token`. This failed when tested with Firefox but worked when tested with Chrome[4]. This is because Firefox and Chrome auto guess the scheme (http) if you manually type `localhost:8080/...` in the address field but for href links and redirects it must be explicitly stated. In short the trusted url had to be http://localhost:8080/token. Also the trusted url sent when authenticating is case sensitive and must be identical to the one entered originally. If the trusted url is set as HTTP://localhost:8080/token but sent as http://localhost:8080/token the server will respond with **The request's 'x_target' value does not match the allowed values for this application. Contact your administrator.**

4.3.3 Sever Time Skew

The function which creates an authenticated url calculates the difference between the local time and the server time and includes this as part of the resulting url. We found that pausing our virtual machine for too long and then trying to make calls to the server resulted in **403**

[4]But only because Chrome let the OS handle the link which was then opened by Firefox as the default browser

Timestamp out of range errors.

4.3.4 PUT Data Format

The sample code at http://docs.valence.desire2learn.com/clients/python/index.html shows a get request as

```
url = uc.create_authenticated_url(route)
r = requests.get(url)
```

so it was assumed that a put request was similar (ie. **requests.put(url,data=...)**) which it it isn't. Specifically a put request looks like

```
r = requests.put(uc.create_authenticated_url(route,
    method='PUT'), json=new_grade)
```

The take away is that the function to create an authenticated url needs to be told the method (ie PUT) and the requests.put function needs to be passed the json data correctly (ie via json=data).

5 Program Installation & Usage

5.1 Requirements

- An operating system that can run Python 3.xxx and the required libraries and has a GUI and a modern web browser.
- Python 3.x and pip installed
- a python virtual environment[5]

5.2 Installation

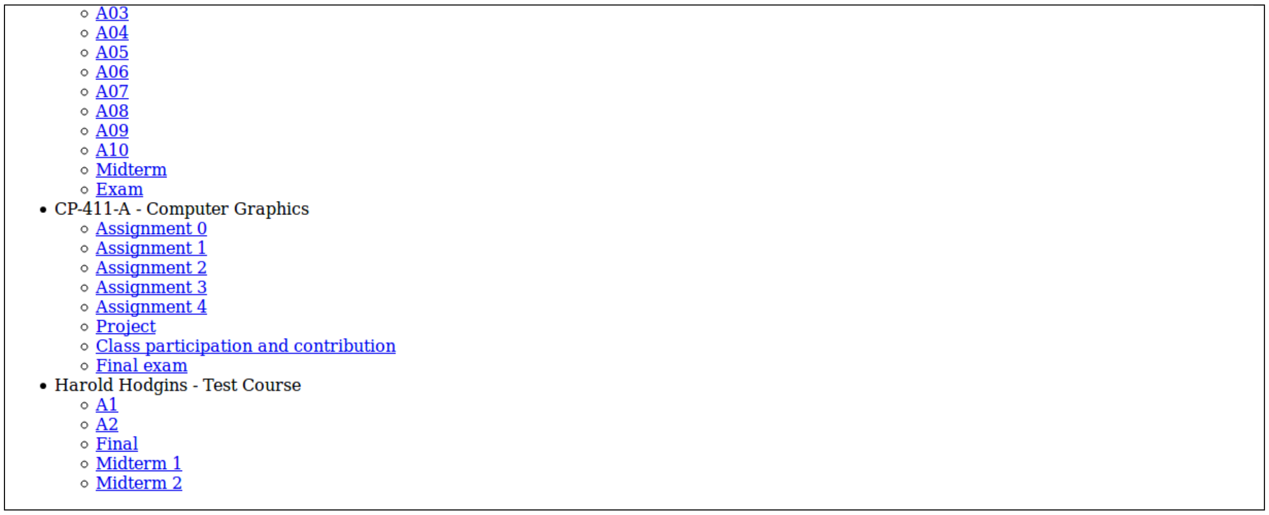
1. Download or clone the source code from https://bitbucket.org/harohodg/brightspace-api-code
2. Create a python 3.x virtual environment (eg 'virtualenv -p /usr/bin/python3.4 API_virtualenv' on Lubuntu 14.04)
3. Activate the virtual enviroment (eg 'source API_virtualenv/bin/activate')
4. Install the necessary libraries (eg 'pip install -r API_code/requirements.txt')
5. Update template_conf_basic.py to use your API credentials and rename the file as conf_basic.py

[5]To keep different version of packages and libraries playing nicely with each other. Not mandatory but very useful. A good tutorial for using virtualenv can be found at `http://python-guide-pt-br.readthedocs.io/pt_BR/latest/dev/virtualenvs.html`

5.3 Usage

To use the application activate the virtual enviroment, change to the source directory and run `python3 setGrades.py`. Once the program is running go to `http://localhost:8080` in your browser of choice and you should be automatically redirected to the login page for the MLS test server.

Once you successfully log in you should see a list of courses and their associated grade items.



- A03
- A04
- A05
- A06
- A07
- A08
- A09
- A10
- Midterm
- Exam

- CP-411-A - Computer Graphics
  - Assignment 0
  - Assignment 1
  - Assignment 2
  - Assignment 3
  - Assignment 4
  - Project
  - Class participation and contribution
  - Final exam
- Harold Hodgins - Test Course
  - A1
  - A2
  - Final
  - Midterm 1
  - Midterm 2

Figure 1: Available Courses

Clicking on a grade item brings up file upload dialog. If you add a file from disk and click `Upload Grades File` you should see one of the following.

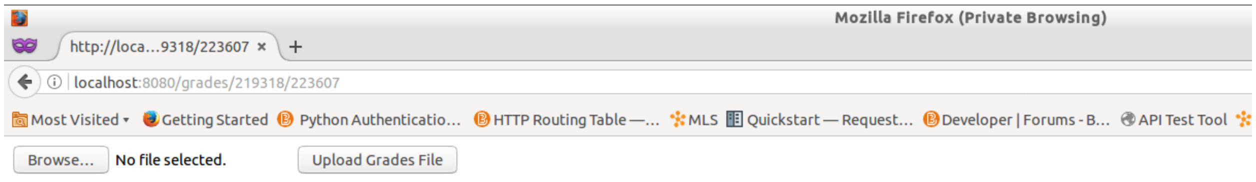
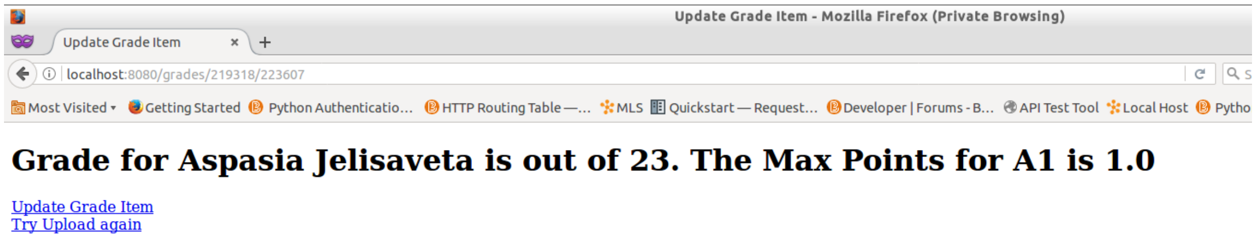


Figure 2: Grades File Upload Page

If the the out of grade for a student does not match the grade item total it will return a warning page with a link to update the grade item total.

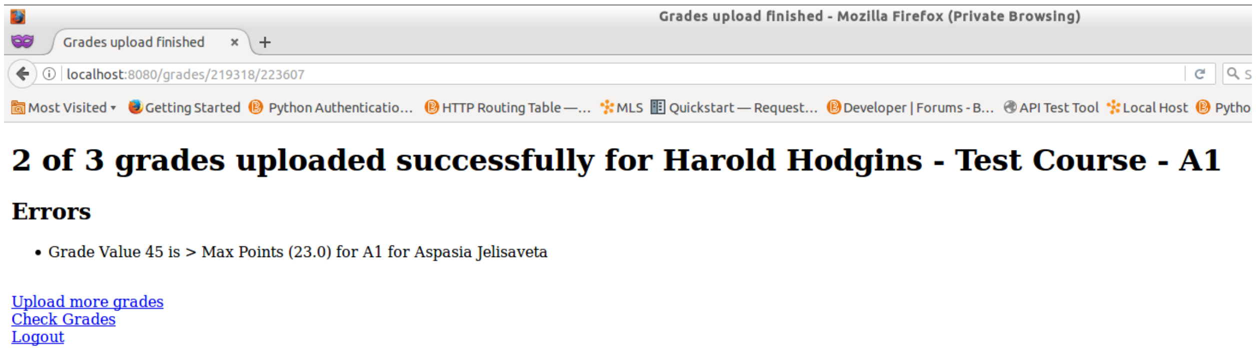


Grade for Aspasia Jelisaveta is out of 23. The Max Points for A1 is 1.0

Update Grade Item
Try Upload again

Figure 3: Invalid Grade

If grades were submitted a page stating how many grades were submitted and lists any failed submissions. Links are provided to go upload more grades, to go check the current grades, or to logout.



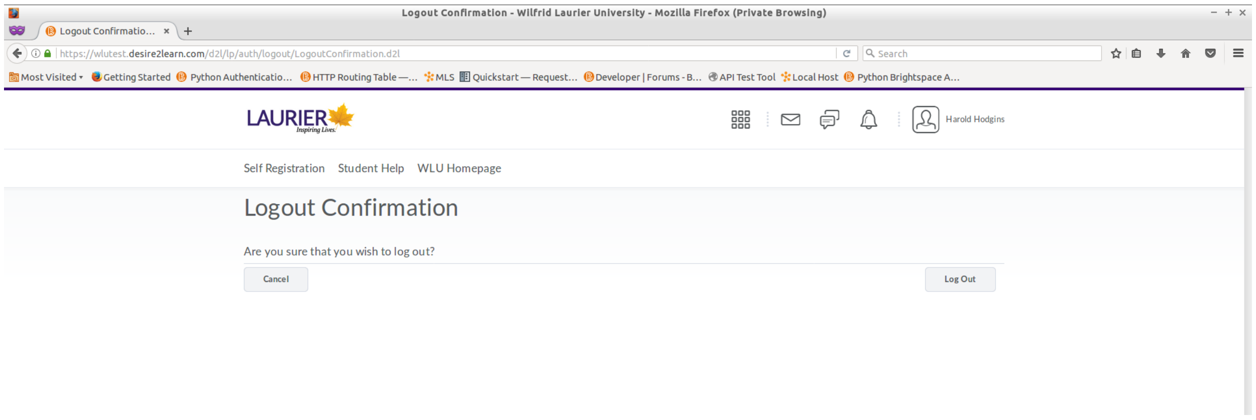
2 of 3 grades uploaded successfully for Harold Hodgins - Test Course - A1

Errors

- Grade Value 45 is > Max Points (23.0) for A1 for Aspasia Jelisaveta

Upload more grades
Check Grades
Logout

Figure 4: Sucessful Grades Upload



Logout Confirmation

Are you sure that you wish to log out?

Figure 5: Logout Page

6 Future Work

6.1 Server Code

The program could be packaged as a docker container. This would ensure that dependencies didn't conflict with the base system, that changes could be easily rolled out, and that any interested parties could easily download and use it with minimal configuration changes.

The program should be updated to support being run on a server and to handle multiple users at once. This would require caching user credentials till they logged out as well as keeping track of who is connected from which browser.

For full deployment at Laurier a new application key and application ID will be needed to associate the app with the production MLS server at Laurier. As well a new trusted url will also be required.

6.2 Grades & Feedback Format

The file format for the student grades and associated feedback should be standardized. It could be done as a csv file using a similar format to the current one used to bulk upload grades with an extra column for feedback.

The final format should use the students Laurier ID # instead of the internal MLS ID #. This means a class list needs to be downloaded and used to translate between the Laurier ID # and the internal ID # that the API uses.

6.3 User Interface

The current user interface although functional is not very nice to look at and by extension not very user friendly. Improvements should be made to guide a novice user through the system and to make the pages look nicer overall. A better home page and an example walk through page could be created (although if the layout is intuitive this wouldn't be needed).

7 Recommendations for D2L

- Fix the Python example code so the sample login works.
- Add more examples to the Python Client Library SDK page. Specifically for doing a PUT request and the various ways of calling the requests library.
- Improve the overall API documentation website layout. Perhaps make all of the Python examples available in one place.
- Add language specific data format examples
- Fix broken URLS so they redirect to their new urls
- Make this project obsolete by upgrading the grades uploading mechanism to accept a feedback column in the csv file

8 References

References

[1] D2L. Brightspace lms for higher education. https://www.d2l.com/solutions/higher-education/. Accessed: 2017-04-23.

[2] D2L. Client sdks. http://docs.valence.desire2learn.com/clients/index.html. Accessed: 2017-09-10.

[3] Harold Hodgins. Brightspace api tutorial. https://cloud.sagemath.com/projects/9501f241-b52e-43f8-9034-7292e8ee54ce/files/tutorial/BrightSpace_API.pdf. Accessed: 2017-04-23.

[4] D2L. Developer platform (march 2017) getting started. http://docs.valence.desire2learn.com/basic/firstlist.html. Accessed: 2017-04-23.

[5] David Boddie. About pyqt. https://wiki.python.org/moin/PyQt. Accessed: 2017-04-23.

[6] Selenium. Selenium webdriver. http://www.seleniumhq.org/projects/webdriver/. Accessed: 2017-04-23.

[7] Brightspace. Sample valence general python. https://github.com/Brightspace/sample-valence-general-python. Accessed: 2017-04-23.

[8] Brightspace. Developer platform (march 2017) python client library sdk. http://docs.valence.desire2learn.com/clients/python/index.html. Accessed: 2017-04-23.

[9] Docker. Docker - build, ship, and run any app, anywhere. https://www.docker.com/. Accessed: 2017-04-23.

9 Acknowledgments

A sincere thank you to the following individuals for their help with this project.

- David Brown for providing the inspiration for this project and for supervising me
- Paul Kleinschmidt for help accessing the API and getting a test course set up.
- Maja Kokotovic and staff at D2L for helping us sort out the how to do a PUT request with the API.
- The Computer Science IAs who where students in the test course.
- All the students who were test subjects for the old versions of the code