The Loop Tool
Installation Guide

Overview

The following learning analytics tool was developed as an outcome of the Closing the Loop project. The Loop Tool provides new opportunities for teaching staff to connect their course design with a set of available analytics visualisations. The Loop Tool is built using Python, Django, Pandas and MySQL and supports both Moodle and Blackboard Learning Management Systems.

Loop Tool Architecture

A key requirement of the Loop Tool was to provide easy visualisations for teaching staff to view course access by content item and LMS tool use. The Loop Tool therefore requires the log files and an export of the course structure from either Blackboard or Moodle LMS. IMS-CP Export files are required from Blackboard and the course export format is required for a Moodle course.

The Loop Tool is made up of the following two components:

- **Data Warehouse:**
  The Loop Tool creates a data warehouse in MySQL. Log and course export zip files are processed and both Moodle and Blackboard data is stored in a single schema. Quiz and forum data is also extracted from the course export zip files. Content and tool items are categorised into Content, Communication and Assessment. The data warehouse implements a star schema design to allow queries by week and day to easily be made. The schema also contains tables that cache all major data components of the dashboards.
- **Dashboard**
  The Dashboard is built with the Django web application framework. The dashboard retrieves cached data from the MySQL database and displays the weekly, overall and content, communication, assessment and overall student dashboards. There are also individual student and content item/tool dashboards.

Course Exports

This section describes the process to export the files required for processing and visualising the respective analytics available through the LOOP tool. The tool requires the LMS access logs and a full course export (including the course structure, forum posts and quiz submissions). Note that the process to export the required files differs across the LMS supported platforms and versions.

Moodle

A course export is required and contains both the access logs and the course structure.

To export from Moodle:

1. Select Backup from the Administration block
2. Select the required information to be included in the backup. Include “grade history” and “Include course logs” must be selected
3. Continue through the wizard. This will result in the generation of a “mbz file” (moodle version of a zip file).

Moodle Hosted

The export may contain some anomalies depending on the version and institutional configuration of the Moodle install. It is recommended that the data integrity is reviewed and validated. In this case of any identified errors you may refer to your IT institutional support and export a csv file directly from the Moodle database.

Blackboard

The IMS-CP course archive file is required and a csv export from the Activity_Accumulator table in the Blackboard database.

The SQL query to obtain the dates in the correct format is below:

```
select PK1
from BB_BB60.COURSE_MAIN
where COURSE_ID = 'coursecode';

--Check the count
select count(*)
from BB_BB60.ACTIVITY_ACCUMULATOR aa
where aa.COURSE_PK1 = 'insertpk';

--Query for the data export
--Change date range as required

select aa.PK1, aa.EVENT_TYPE, aa.USER_PK1, aa.COURSE_PK1, aa.GROUP_PK1,
aa.FORUM_PK1, aa.INTERNAL_HANDLE, aa.CONTENT_PK1, aa.DATA,
       to_char(aa.TIMESTAMP, 'DD-MON-YY HH24:MI:SS') TIMESTAMP,
       aa.STATUS, aa.SESSION_ID, aa.MESSAGES
from BB_BB60.ACTIVITY_ACCUMULATOR aa
where aa.COURSE_PK1 = 'insertpk'

      and aa.TIMESTAMP
      between to_date('11-12-2015 00:00:01', 'DD-MM-YYYY HH24:MI:SS')
      and   to_date('13-12-2015 23:59:59', 'DD-MM-YYYY HH24:MI:SS')
      ;
```

Loop Installation and Configuration

The source code is available as open source via the Github repository at: https://github.com/looptool/download

1. Clone the repository
   ```
   $ git clone https://github.com/looptool/download.git
   $ cd download
   ```
2. Install the project requirements
   ```
   $ pip install -r requirements.txt
   ```
   A Virtual environment can be created for this project.
   `A Virtual Environment` is a tool to keep the dependencies required by different `projects in separate places, by creating virtual Python environments for them.`
3. Setup a MySQL database using the datawarehouse/cloop_olap.sql script. This is the OLAP database.
4. Enter the database settings in cloop_project/settings.py. A secret key also needs to be entered. There is a local database for the Dashboard (this can be in any database supported by Django) and the OLAP database (currently in MySQL).
5. Setup the local django database
   ```
   $ python manage.py makemigrations
   $ python manage.py migrate
   ```
6. Create an admin user
   ```
   $ python manage.py createsuperuser
   ```
7. Integrate Django with a Webserver. Apache or Nginx can be used. Instructions for Apache are included.
   Create a Virtual Host:
   ```
   $ mkdir  -p ~/public_html/loop
   $ sudo chmod -R 755 /home/ubuntu/public_html
   $ cd ~/public_html/loop
   ```

   Edit the default virtual host file
   ```
   $ sudo nano /etc/apache2/sites-available/000-default.conf
   ```
   The main things to change are:
   - comment out the document root
   - add the WSGIScriptAlias to point to the location of the wsgi file
   - Include the <Directory> directive otherwise there will be apache access errors. Without <Directory> you will get an `H01630: client denied by server` error.
   - Includes a mapping for a static folder for the django project (i.e. for serving static files for images, js or css)

     <VirtualHost *:80>
     # The ServerName directive sets the request scheme, hostname and port that
     # the server uses to identify itself. This is used when creating
     # redirection URLs. In the context of virtual hosts, the ServerName
     # specifies what hostname must appear in the request's Host: header to
     # match this virtual host. For the default virtual host (this file) this

```
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        #DocumentRoot /var/www/html
        WSGIScriptAlias / /home/ubuntu/public_html/loop/loop_project.wsgi

        Alias /static/ /home/ubuntu/public_html/loop/loop_project/static/
        <Location "/static/">
            Options -Indexes
        </Location>

        <Directory /home/ubuntu/public_html/loop>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Creating a WSGI File

The [Web Server Gateway Interface] (WSGI) is an interface that maps requests from a Webserver (e.g. Apache) to a web application in python.

A WSGI configuration file `/var/www/loop_wsgi.py` must be setup. In the file a virtual environment can be specified along with the django project settings:

Create the .wsgi file mentioned in `/etc/apache2/sites-available/000-default.conf`:

```
$ sudo nano ~/public_html/loop/loop_project.wsgi
```

Enter the following:

```
import os
import sys
sys.path.append('/home/ubuntu/public_html/loop/loop_project')
os.environ['DJANGO_SETTINGS_MODULE'] = 'loop_project.settings'
import django.core.handlers.wsgi
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Reload Apache VirtualHost and Server

```
$ sudo a2ensite
$ sudo /etc/init.d/apache2 reload
```

Note: Use a2dissite to remove a loaded virtualhost

8. Go to the /admin in a Web browser and add users and courses (adding *a course repeating event* is mandatory). You will need to login using the admin username and password that was created in step 6. You must also set up the dates for key course events. Users are added and assigned to courses using the admin interface.
9. Setup the data warehouse config file (datawarehouse/config.json). The course_id must match the course_id from the Django dashboard database (the one automatically generated by Django, not the ID defined by the user of the system). The format for the json file is below:

   [
   {"course_id": 5, "start_date":"27/JUL/15", "end_date":"06/NOV/15", "course_type": "Blackboard"},
   {"course_id": 6, "start_date":"27/JUL/15", "end_date":"06/NOV/15", "course_type": "Moodle"}
   ]

   Note: Semester start and end dates must be included.
   In datawarehouse/build_olap.py enter the database access credentials.
10. Create a /data directory. Each course must have a folder with the course_id as the name. The course export file must be unzipped and placed in the folder. The log file must also be placed in the folder and named log.csv.
11. Run datawarehouse/build_olap.py to create the data warehouse and cache the display results for the dashboard.
12. All users will now be able to login to the dashboard.
13. The OLAP DB can be started.
14. The server can be started

    ```
    $ python manager.py runserver
    ```

This document is part of the "Completing the Loop" Handbook, which can be downloaded from http://melbourne-cshe.unimelb.edu.au/completing-the-loop . Please cite as:

Corrin, L., Kennedy, G., de Barba, P., Lockyer, L., Gašević, D., Williams, D., Dawson, S., Mulder, R., Copeland, S., & Bakharia, A. (2016). *Completing the Loop: Returning Meaningful Learning Analytic Data to Teachers.* Office for Learning and Teaching.