Infandango: automated grading for student programming

Hull, M., Powell, D., & Klein, E. (2011). Infandango: automated grading for student programming. In ITiCSE '11 Proceedings of the 16th annual joint conference on Innovation and technology in computer science education . (pp. 330). ACM Association for Computing Machinery. 10.1145/1999747.1999841

Infandango: Automated Grading for Student Programming

Mike Hull
School of Informatics
University of Edinburgh, UK
m.j.hull@sms.ed.ac.uk

Dan Powell
School of Informatics
University of Edinburgh, UK
d.c.powell@sms.ed.ac.uk

Ewan Klein
School of Informatics
University of Edinburgh, UK
ewan@inf.ed.ac.uk

ABSTRACT

Infandango[1] is an open source web-based system for automated grading of Java code submitted by students. Uploaded Java files are compiled and run against a set of unit tests on a central server, with results being stored in a database. Students gain near-instant feedback on the correctness of their code, and instructors are able to monitor the progress of students in the class.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education

Keywords

Automatic grading, Java, Django, reStructuredText, Sphinx

1. OVERVIEW

Like many other educational institutions, we run an introductory course in Java programming for undergraduates, and face the problem of giving students feedback on their answers to practical exercises in the lab. Although this work is, in the main, formatively assessed, we have found that students appreciate receiving a numerical grade for their work; the obvious mechanism for achieving this is to run relatively fine-grained JUnit[2] tests over the code. In developing our own software for automatic assessment, we focussed on the following requirements: **modularity**, **security**, and **ease of use** for both students and instructors. We based our software on the Django[3] web framework because of its excellent support for agile development.

Figure 1 shows the architecture of infandango. Communication between the web frontend and Jester, our Junit testing daemon, is mediated by a PostgresSQL[4]database. The database holds information about files that have been submitted by users and the results that they receive from Jester. User access to the system is protected by CoSign[5] authentication. In addition, there is a set of Python tools for uploading question sheets, batch uploading code for testing by Jester, and for generating reports.

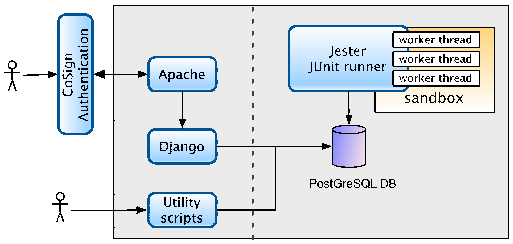

Figure 1. The infandango server

When a Java source file is submitted, Jester compiles it, executes it in a sandbox and runs the unit tests. If the tests succeed, the student is awarded marks in the database; if not, they are informed about which tests failed.

Lab exercise are written using the Sphinx[6] extensions of reStructuredText.[7] This provides excellent support for syntax highlighting of code blocks, and made it very easy to produce standalone versions of the lab exercises in addition to those served *via* infandango.

2. DEPLOYMENT

Infandango was first deployed in January 2011 for our introductory Java programming course. Exercises were written on a weekly basis and merged into the live system, resulting in a total of 45 exercises marked by over 150 JUnit tests. Over 100 students used the system, submitting close to 5,500 Java source files, and to date, over 20,000 unit tests have been executed on student code.

3. CONCLUSION

The most notable features of infandango are its modularity and extensibility. Components of the system are loosely coupled: standalone modules for authentication, web frontend and unit testing interact with each other only where necessary. Since the CoSign authentication uses standard HTTP headers, it can easily be replaced by another method. The frontend could be replaced or further customised without affecting the tester in any way, and in principle, different test components could replace Jester. Finally, there is little dependence on PostgreSQL, and switching to another database manager would take very little effort.

[1] https://bitbucket.org/ewan/infandango
[2] http://www.junit.org/
[3] http://www.djangoproject.com/
[4] http://www.postgresql.org/
[5] http://cosign.sourceforge.net/

[6] http://sphinx.pocoo.org/
[7] http://docutils.sourceforge.net/rst.html

ITiCSE'11, June 27–29, 2011, Darmstadt, Germany.
ACM 978-1-4503-0697-3/11/06.