Web-based program compilation on Moodle

Kartik Modi
Department of Computing
Macquarie University
Sydney, Australia
Eng.kartik@gmail.com

Abstract

This paper presents an online Java compilation plug-in for the Moodle Learning Management System (Moodle Community, n. d.). The system allows students to compile Java programs directly through the Moodle Learning Management System so that they can concentrate on the programming concepts rather than learning to operate new technologies. The system also provides auto marking features and the ability to collect data for analyzing students programming behavior. The technologies and Java APIs related to the online compilation processes are discussed in detail. The document also highlights security problems related to the servers and their resolutions. Development outcomes are presented and opportunities for future work are discussed.

Keywords
Online Java Compiler, Moodle plug-in

1 Introduction

Most universities across the world are incorporating web-based Learning Management Systems into their courses. Moodle (Moodle Community, n. d.) has been rated as the most popular Learning Management System (Kathy D.Munoz, Joan Van Duzer, 15 February, 2005) and comes with a number of features for creating forums, organizing quizzes, giving choices to students, conduct surveys, uploading assignments, holding chat sessions, creating workshops and much more. Importantly, Moodle is a modular, open source system and as such allows universities to customize the Moodle server for their usage.

One problem that students experience when first learning computer programming is cognitive overload caused by not only having to learn programming concepts but also how to operate an Integrated Development Environment (IDE). As well, teachers experience problems coordinating coursework tasks in terms of specifying tasks, collecting student responses, analyzing student work and providing feedback.

This paper reports on the development of a new Moodle JavaOnline question type plug-in that provides the facility to compile Java programs online. This means that students do not have to install and configure different compilers on their machines. As well, the plug-in reduces the overhead of evaluating answers and provides error statistics to teachers which can help them to identify more common mistakes made by students. The plug-in is developed considering security aspects and runs independently from the Moodle server.

After briefly discussing related work, this paper outlines the overall design of the module in terms of the way in which the PHP/Java Bridge (Jost Bökemeier, n. d.) is used to communicate between the Moodle server and Java compiler. The approach to information processing is explained and security issues are identified. The overall outcomes of the project are presented and areas for future research and development are proposed.

2 Related Work

Online compilers for languages like C, C++, PHP, Ruby are available but all of these compilers have limitations and they are not design to integrate with Learning Management systems. Considering Java programming language, there are number of Java Interpreters developed including JOSH (Stephan Diehl & Claudia Bieg, 2008), MiniJava (E. Roberts, 2001), DynamicJava (S. Hillion, n.d.) and BeanJava (P. Niemeyer, n.d.). These Interpreters are developed in early days to emphasize on the same problem as online

compilation but these approaches define rule engines and complex implementation. User programs pass through this rule engines and evaluate it. However, compilers and interpreters have different set of operations and now programming languages like Java provides rich sets of APIs using that online compilation problem can be simplified.

Until recently no Learning Management System provided a complete solution for online Java compilation. In the past few months the Junit question type (Süreç Özcan, January 2009) plug-in was released for the Moodle server, which compiles Java programs online. This plug-in uses the exec() method to compile and execute Java Programs.

However, there are limitations associated with Junit's use of the exec() method, such as the limited size of the output buffer and a range of security issues. The main purpose of exec() method is to execute commands over the shell and return back a result of execution but compilation of Java programs is different. Passing small Java programs to exec() method may compile properly in some cases but not all. For instance a Java program which does extensive string operations may throw exceptions such as ArrayIndexOutOfBoundException, NullPointerException and in most of cases the exec() method shows blank output or possibly throws a stack overflow exception. This is because of the limited buffer size.

In terms of security issues, the Junit depends on the Java policy file setup by the administrator and if setup incorrectly can impose security vulnerabilities. As well since the Junit plug-in must call the exec() method on the same server as the Moodle server there is the potential that executed code may not only crash the plug-in but also the entire Moodle server.

The approach adopted in the JavaOnline question type utilizes an entirely different design in order to the allow compilation of Java programs using Java API's and on a different server. This provides a more secure and robust approach to online compilation of Java programs.

3 Technology Used

The JavaOnline question type plug-in is implemented using the API's of Java Standard Edition 6 (Sun Microsystems Inc, 2009c). New API's available in Java 6 make it possible for programs to compile source code and these programs can reside on a remote server other than the Moodle server. The PHP/Java Bridge provides a communication channel between the Moodle server and the Java web-server, as shown in Figure 1.

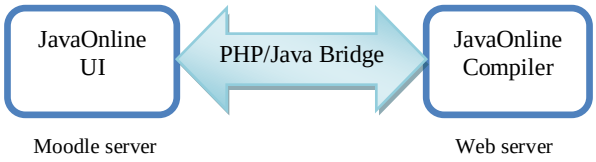

Figure 1 – Structure of JavaOnline question type

3.1 Moodle Learning Management System

The Moodle Learning Management System is designed and developed in PHP. Moodle has well defined and organized structures for each of its modules. There are different types of questions, like multiple-choice questions, essay questions, description question and numerical questions. These are referred to as "question types" in the Moodle Learning Management System. Developers can use Moodle development templates that are available on Moodle web-site to develop new plug-in for the Moodle sub-system.

3.2 JavaOnline Question Type Plug-in

The JavaOnline question type uses the core libraries provided by the Moodle environment. Each of the question type modules in the Moodle uses the same folder structure. The root folder for each module has the same name as question type. Under this folder, it has db folder that contains information about schema for that module and lang folder which contains information about languages. Other than that it also contains at least two most important files; edit_MODULENAME_form.php and questiontype.php. These two files represent user interface for JavaOnline question type. The Folder structure of JavaOnline is shown in Figure 2 below.

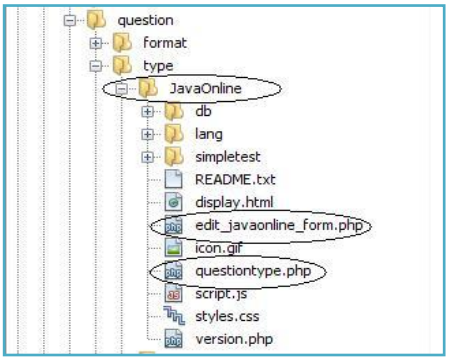

Figure 2 - JavaOnline plug-in on the Moodle server

JavaOnline provides two different views one is for teachers and another for students, as is the

case for all question types in the Moodle. The teacher view allows teachers to create new JavaOnline questions by putting information about question. This information gets stored In the Moodle database.

In the student view, students are provided with two text areas and two buttons. In the first text box area they can write Java programs and in another, they can see the output. The buttons enable them to compile and execute Java programs.

Each attempt to run JavaOnline question by the students gets stored as unique session and students are allowed to compile up to maximum numbers of attempts set by teachers. If teacher has set unlimited attempts then student can compile any number of times. In the JavaOnline question type, students' responses are stored in $state->responses. Each response is in the form of a string array that holds the Java program written by the student.

3.2.1 Java Standard Edition 6

Java SE 6 (Sun Microsystems Inc, 2009c) is the current major release of the Java SE platform. In this release they have introduced new package javax.tool that offers interfaces for retrieving an compiling files passed as strings. Thus clients can locate and run Java compilers from within a program. The package also provides other interfaces that generate diagnostics and override file access in order to support the compilation process (described below). Figure 3 shows javax.tool interface.

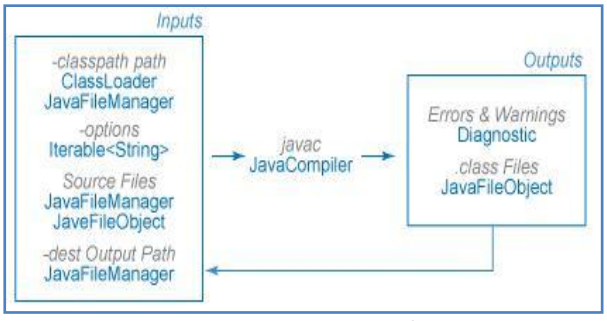

Figure 3 – javax.tool Interface

3.2.1.1 javax.tool.JavaCompiler Interface

The new JavaCompiler interface of the tool package invokes Java programming language compilers from programs. The interface can create Java classes by compiling valid Java source, bypassing the previous need to validate byte code, or understand new object models of classes, methods, statements and expressions (David J.Biesack, Dec, 2007). It simplifies and standardizes the compilation process by providing one supported mechanism for code generation and loading that is not limited to passing a single file of source code.

3.2.1.2 javax.tool.Diagnostic Interface

The Diagnostic Interface usually reports a problem at a specific position in a specific file. It supplies error details like the line number, column number, start position, end position, error message, source of error and kind of error. It also provides details about warning and information messages generated during compilation.

3.2.1.3 javax.tool.FileObject Interface

The FileObject Interface is an abstraction tool. In this context file means abstraction of regular files and other sources of data. For example, a file object can be used to represent regular files, memory cache or data in database. All methods in this interface might also throw a securityException if security rules are violated.

3.3 PHP/Java bridge

As JavaOnline question type plug-in developed using Java and PHP language, communication between these two different languages was major concern. This problem is solved using PHP/Java Bridge (Jost Bökemeier, n. d.). PHP/Java Bridge is an implementation of a streaming XML-based network protocol, which can be used to connect a native script engine with the Java virtual machine. This API works under Java and PHP enable servers. The following example shows the important library file that needs to be included in the PHP file to create the Java objects using PHP.

```
re-
quire_once("http://localhost
:8080/Java/java/Java.inc")
```

The following examples show creation of Java objects, methods and a session from PHP.

```
/* invoke ja-
va.lang.System.getProperties
() */
$pro=java("java.lang.System"
)->getProperties();

/* convert the result object
into a PHP array */
$array = ja-
va_values($props);
foreach($array as $k=>$v) {
```

```
echo "$k=>$v"; echo
"<br>\n";
}

/* Create Java session */
$session = java_session();
```

3.4 Justification of the approach

There are other approaches like Runtime.exec() on Java and exec() on PHP, that can be used to compile and execute Java programs. These approaches are simpler but do not provide a complete solution to online Java compilation. Some of the issues related with exec() are; it does not provided detail compilation statistics, the programs need to be compiled on the same machine, with larger error statistics it may throw statckoverflow exceptions and other unexpected results.

existing frameworks such as JSP (Java Server Pages) already support this capability.

4 Information Processing

For the purposes of this paper, information processing describes information that is passed from the JavaOnline user interface to the webserver where the Java programs get compiled. Figure 4 gives overview of information processing for the JavaOnline question type.

After initiating the session for the JavaOnline question type, students are provided with compile and execute buttons on their screen. Only after a successful compilation Will the execute button become enabled.

When students compile their Java program using compile button on the user interface,

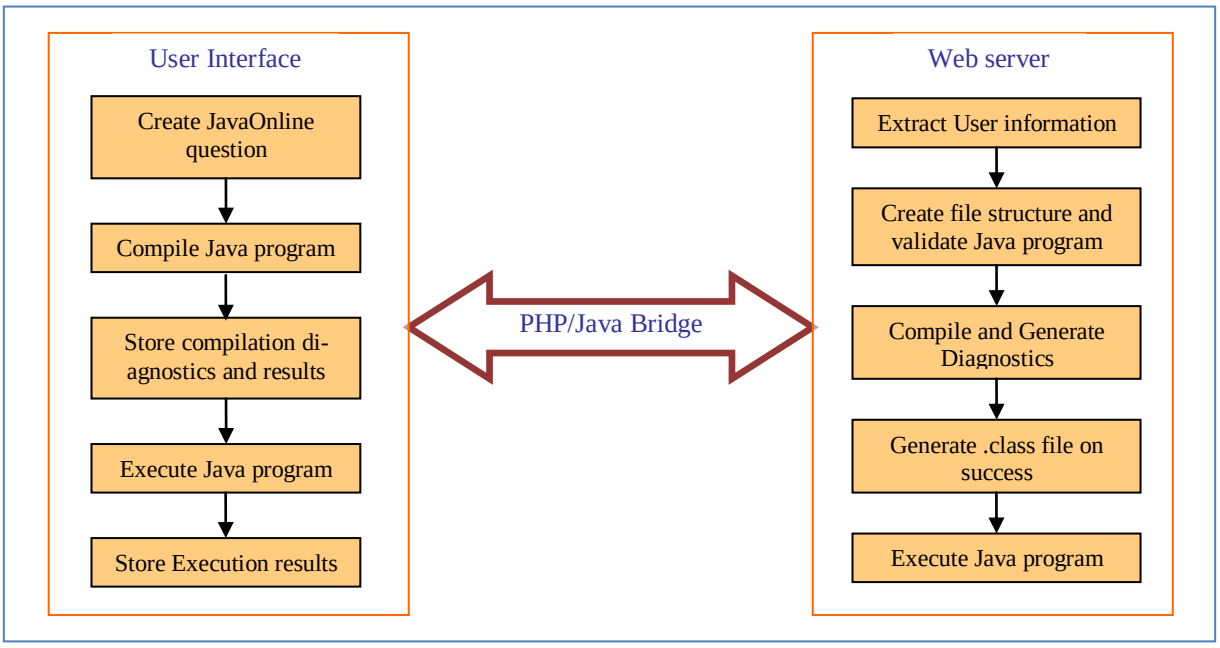

Figure 4 Information processing in the JavaOnline question type

On the other hand, APIs of Java SE 6 provides the opportunity for a more robust and reliable solution. It runs on the same JVM (Java Virtual Machine) as the current program which reduces the overhead of starting a new JVM. As well, the complier keeps running on the server so when more source files are compiled the compiler does not need to be reinvoked each time the way it would if using exec() (Roedy Green, February, 2006). The JavaCompiler works with any platform, whereas platforms need to be individually configured to use the exec() approach. The idea of extending applications via compiling and loading Java extensions is not new, and several

communication with web server is initiated using the PHP/Java Bridge. The information present in state->responses gets converted into a Java string object from the PHP array. Information such as the user id, user name, default file path set by the administrator, number of attempts, the current attempt number and the Java string object are then passed to the CompileMe() function on the web server. Note that the Java string object is nothing but the Java program.

The CompileMe funtion calls the filesystem method which manages the directory structure on the web server. It uses the path defined by the administrator as the root directory. It creates

directories for each user by using their user id under the root directory. Whenever students compile their Java programs the CompileMe function checks for proper file structure and generate class files under that directory only.

The CompileMe() function uses the Java APIs to compile the student's Java program, and generates error diagnostics on a compilation error. On unsuccessful compilation, it passes back diagnostics to the user interface and displays them on the screen. On successful compilation, the diagnostics object has a null value which triggers the execute button to become enabled on the user interface.

If students have used all attempts then the compile button becomes disabled. For instance, consider the scenario, where teacher has set the number of attempts to be two. If the student has compiled program correctly on first attempt then execute button gets enabled for execution but for the second attempt compilation fails which in turn causes the compile and execute buttons to become disabled. Each of these compilation results gets stored in the database.

When students click on the execute button, the same Java session created by PHP/Java Bridge is used to communicate with the web server. On execution, the ExecuteMe() method is called which uses the Runtime.exec() method to execute the generated class file. The result of execution is sent back in the form of a string to the user interface. In the case of a runtime exception, exeception details are pass back to the user interface.

5 Outcomes

The current version of the JavaOnline question type is capable of performing all of the functions outlined in the above design description. This includes the capacity to:

- Provide the teacher and student interface
- Allow teachers to define a question
- Allow students to attempt the programming problem online by performing successive compilations
- Have error and other diagnostic messages returned to the student
- Store the results of each attempt into the Moodle database
- Execute the Java program.

Thus the JavaOnline plugin achieves the requirements of the initial design specification.

6 Security

Security of a server is major concern when considering online compilation. Online compilation on the Moodle server means allowing users to compile and run Java programs on the server. A Java program may be written by a beginner or someone who is experts in the Java programming. Both scenarios have a potential threat to the server. In the case of expert students, they may try to program using different Java API's which may not be configured or protected in the security policies set up by the Administrator. In the case of normal students, they may not be aware of exception handling or memory management. For example consider the situation where teacher has asked for a program which does the simple file operation of copying content of one file to another file. In this case if student put extra loop in their program which just keeps making files on the server then the server will run out of memory and eventually crash.

As well, Moodle is a web based which means that multiple users can access it at the same time and as such may create problems related to resource sharing. These problems can potentially lead to deadlock, and thus process management needs to be considered during configuration of the server. This might be one of the reasons that none of the online learning systems have implemented web-based compilation.

The JavaOnline plug-in always checks for size of directory allowed for each student before compiling. If this size reaches to maximum allowed size then they get error message and JavaOnline plug-in disables compile and execute buttons. In the JavaOnline plug-in, students are only allowed to access their directory. They do not have access outside their directory.

The security issues related with the servers are discussed in detail on the Java Security Overview (Sun Microsystems Inc, 2005a) and Moodle Administrator Document (2009). There are a few methods defined to minimize threats on the servers and help to achieve goals of online compilation. It is responsibility of the administrator to configure Moodle and web server properly. Java provide security framework that uses policy files. This policy files contain configuration related access of API's available in Java. This framework restricts unauthorized users.

In the JavaOnline question type security is provided by defining security policy. This policy file kept on the web server. On start up of the web server this file gets loaded. An example of the policy file is given below.

```
//example of policy file
grant codeBase
"file://loction/JavaOnline.j
ar" {
  permission ja-
va.util.PropertyPermission
"user.dir", "read";
  permission ja-
va.io.FilePermission "${us-
er.dir}${/}-",
 "read,write,delete";
};
```

7 Future work

There are many opportunities to further enhance the JavaOnline question type plug-in. For instance, the implementation of automatic evaluating and grading has not yet implemented. There is also the potential to allow teachers to set up practical Java tests on the Moodle Learning Environment. Teachers could create test of multiple Java questions that students would have to answer within a time limit. The system would be able to evaluate answers based on input and output criteria defined by the teacher and generate result at the end of the test.

As well the conceptual design underpinning the JavaOnline plugin could be used to create Java projects online. Multiple students could work in single group on a single project. from anywhere, anytime without downloading and installing IDE tools to their machine.

8 Conclusion

The JavaOnline question type plug-in for Moodle LMS presented in this paper provides a solution for online compilation of Java source code. This system enables students to compile their programs without having to configuring their machine for Java program compilation. It also generates detailed statistics of student's compilations that can help teachers to improve their teaching methodologies. The JavaOnline plugin also uses secure and robust approach to online Java compilation that overcomes limitations in the design of the JUnit question type.

Acknowledgments

I heartily thank my advisor Matt Bower for his enormous help and support.

References

Douglas Kramer, 1996, *The Java Platform: A white paper*, viewed March 23, 2009, http://java.sun.com/docs/white/platform/javaplatformTOC.doc.html

David J.Biesack, Dec, 2007, *Create dynamic applications with Javax.tools*, viewed on May 19, 2009, http://www.ibm.com/developerworks/java/library/j-jcomp/index.html

E.Allen, R.cartwright and B. Stoler, n.d., *DrJava*, viewed March 22, 2009, http://drjava.sourceforge.net

Moodle community, n. d, *Moodle.org:open-source community-based tools for learning*, viewed March 21, 2009, http://moodle.org/

Moodle Administrator document, 2009, *Administrator document*, viewed March 26, 2009, http://docs.moodle.org/en/Administrator_documentation

Kathy D.Munoz, Joan Van Duzer, 15 February, 2005, *Blackboard vs. Moodle A Comparison of Satisfaction with Online Teaching and Learning Tools*, viewed May 18, 2009, http://www.humboldt.edu/~jdv1/moodle/all.htm

P. Niemeyer, n.d. *BeanShell*, viewed March 23, 2009, http://www.beansheell.org/

Roedy Green, February, 2006, *JavaCompiler: Java Glossary*, viewed on May 19, 2009, http://mindprod.com/jgloss/javacompiler.html#BENEFITS

Sun Microsystems Inc, 2005a, *Java Security Overview*, viewed March 26, 2009, http://java.sun.com/developer/technicalArticles/Security/whitepaper/JS_White_Paper.pdf

Sun Microsystems Inc, 2008b, *Java Platform Standard Edition*, viewed March 25, 2009, http://java.sun.com/javase/6/docs/api/javax/tools/package-summary.html

Sun Microsystems Inc, 2009c, *Java SE 6*, viewed May, 20, 2009, http://java.sun.com/javase/6/

Süreç Özcan, January 2009, *Junit question type*, viewed April 25, 2009, http://docs.moodle.org/en/Junit_question_type

Jost Bökemeier, n. d., *Php/Java Bridge*, viewed April 20, 2009, http://php-java-bridge.sourceforge.net/pjb/index.php