Tool to Automatically Generate Database Wrapper Classes

Prepared by:
Jason Kelly
Department of Computer Science
The University of Waikato

Prepared for:
Sally Jo Cunningham
Department of Computer Science
The University of Waikato

This proposal outlines the development of a software tool that would save developers time when writing database wrapper classes. When writing applications that rely on retrieving and maintaining data in a database system, developers usually write a set of reusable classes which contain all the database-specific code. Often, especially with large databases, these classes are repetitive and tedious to write. Often, because much of the code is fairly similar, developers may copy, paste and edit functions in the code, which greatly increases the chances of bugs in the code. This proposed tool would examine the Structured Query Language script file that was used to create the database, and based on the schema information, would then generate a set of wrapper classes that could be used to access the database tables. This tool would reduce the amount of developer time spent writing and debugging wrapper classes.

Introduction

In the development of information-centric applications, it is often common to embed Structured Query Language (SQL) statements within a series of reusable wrapper classes that the application logic can use for the retrieval and management of information stored in a database system. For large and complex database schemas, the wrapper classes can often be very tedious and repetitive to write. A solution to this, which this project proposes, would be to build the necessary wrapper classes automatically based on the database schema script file used to build the database.

Background

As software applications become more information-orientated, their reliance on information storage and retrieval systems grows. The demand for information storage and retrieval systems resulted in the development of database systems. While many database systems, both proprietary and open-source, have matured and become very powerful within the domain of managing information, the technology for interacting with the database systems programmatically is often complex to implement, such as with the *Open Database Connectivity* (ODBC) *Application Programming Interface* (API) (Grechanik *et al.*, 2002). Access is not standardised across the different database systems which offer different levels of ODBC compliance.

For these reasons it is better to offer the application developers a level of abstraction from the database access code by placing all database access code in a group of reusable classes in a centralised location. This, as Grechanik *et al.* (2002) also points out, helps to increase code maintainability.

Design

The proposed system will be written in the Java programming language and will thus be operating system (OS) independent to a degree – it will require a Java Runtime Engine (JRE) or other suitable Java Virtual Machine (JVM) to be installed on the computer it is running on. The proposed tool will be written using Java 5.0

language constructs, and will require a JRE of version 1.5 or greater to run. The source code generated by this tool will be compatible with a JRE of version 1.4 and greater.

The tool should run as an Eclipse IDE plug-in; however, if research shows that making it an Eclipse plug-in is too costly time-wise, then it will be developed to run as a stand-alone Java application.

It will accept, as its input, any valid SQL script file that contains SQL Data Definition Language (DDL) which creates a database and one or more tables in the database.

The system's output will include the Java source code, written to disk, for the wrapper classes which are generated from the SQL DDL script file given as the input. Several types of wrapper classes will be examined, and the one that is most intuitive and offers the best ease of maintenance will be used.

Evaluation

As this tool is intended to be used from within the Eclipse IDE, it will offer developers using Eclipse as their development tool an easy way to generate wrapper classes. This would, however, prevent other developers who do not use Eclipse from using this tool. As a stand alone application, this limitation would be avoided; however, Eclipse developers would not have the convenience of having this tool as a part of their IDE.

This tool will generate wrapper classes written in Java. Although the tool will be written to support the ability to write in different languages, for the purposes of this project only Java output will be supported.

Proposed Schedule

The table below outlines the proposed deadline dates for this project.

Task / Component	***Completion***
Research possibility of using Eclipse plug-in architecture	30/03/2006
Research SQL script parsers	2/04/2006
Implement SQL script parser	15/04/2006
Design plug-in architecture of code writer for future expansion	1/05/2006
Deliverable: Interim Report	2/06/2006
Build Java Code Writer	2/06/2006
Test and refine Java Code Writer	20/07/2006
Conference presentation draft	16/08/2006
Deliverable: Conference abstract	18/08/2006
Conference	1/09/2006
Deliverable: Final report	11/10/2006

Resources

The hardware and software resources required to develop this project are readily available. For the development of this project a computer with Java SDK versions 1.4 and 1.5 will be used. For writing and performing initial testing of the code I will use the freely available Eclipse Integrated Development Environment (IDE). I will test the generated Java source code against a number of freely available database servers, such as MySQL, PostgreSQL, and Microsoft SQL Server 2005 Express.

There already exists an open source tool which performs a similar function as would the tool this project aims to develop, although upon an initial examination it seems that this tool generates Java source code for use in enterprise web applications. Parts of the open source tool may prove to be useful in this project, so the open source tool will be examined further.

Two utilities I have developed previously will be useful in the development of this project. The first utility generated C# source code for database table wrapper classes. It is a simple utility, written in Java, which was used to write and comment the getter and setter methods for wrapper classes of large tables (25 or more columns). The second utility, written in PHP, is a web-based tool for editing database table data. The utility dynamically builds the HTML form, and interacts with the database, based solely on an XML description of the database. If changes were made to the database, or a different database was used, all that would be required for the utility to work would be to update the XML database description.

Conclusion

This project, when completed, will provide developers with a time-saving tool that can be used to quickly generate a series of database wrapper classes based on the SQL DDL script file defining the database. The generated code will be usable immediately after generation, that is, it will retrieve data from and save data to the database without any modifications to the code. The code will also be fully documented and easy for the developers to customise should they require any code customisation.

References

Grechanik M., Perry D. and Batory D. (2002). “An approach to evolving database dependent systems.” *Proceedings of the International Workshop on Principles of Software Evolution*, Orlando, Florida, pp. 113 – 116.