Generator-GUI, an easy and powerful java code generation tool

Project 6: Di LIN 57651410, Peiling LIU 57647795

Abstract—Generator (GitHub: https://github.com/GreedyStar/generator) is a tool for automatically generating Java code based on database tables. However, during our use of the tool, we found that there is still lots of space for improvement. The existing tool requires developers to install related dependencies, set parameters, set templates, write main function and a series of tedious operations to generate the codes. We simplify the code generation process by developing a GUI for it and the developer only needs to operate on the GUI to generate code. The original tool could only generate code in a certain project path. By adding file-related functions, developers can generate code in any path. We also implemented some additional functions, such as allowing developers to select tables and files to generate code, saving, and importing data settings, testing database connections, and more. As a result, our improved code generation tool is simpler and more powerful, and the generated code is error free.

Index Terms—Generator, GUI, Java, Spring boot, Mybatis, Freemarker, JavaFX, Mysql, File

———————————— ◆ ————————————

1 Introduction

Java is a widely-used cross-platform language. As is well known, massive applications and software are based on java frameworks, such as Spring, Spring MVC, Spring boot, and Mybatis. However, when we use java frameworks to develop something like a simple blog web application based on Vue and Spring boot, we often waste lots of time writing repetitive and boring codes, such as POJO (Plain old Java object) and CURD interfaces.

A well-designed code generation tool can help ease the problem and make developers more focused on technical details. Besides, we can use a code generation tool to keep the code format consistent in the case multiple developers participate in the code writing. For example, when we want to develop an article management module based on the blog application, we need to write several format files to achieve simple CURD functions with our own hands. As an alternative, we can use the tool to generate some template codes and avoid the tedious steps.

However, there is still some inconvenience in the existing tool (Generator: https://github.com/GreedyStar/generator) for our requirement. Problems concerning dependency, usability, and flexibility can prevent users from developing efficiently. For instance, we use the existing java code generation tool to generate required files for a demo blog project. To do this, first, we must import all related dependencies about the existing tool. Then we need to set global parameters and create a configuration file according to the document. After this, we should import a code template to define the code format, followed by a test class and a function to invoke the tool. Finally, we can check the files generated by the function if everything goes right. Otherwise, we need to debug errors by ourselves.

To address the limitations of existing tools, we present our solution: Generator-GUI, an easier and more powerful java code generation tool based on the existing tool. This tool allows users to easily generate entity classes, mapper classes, and MVC-related codes. The tool is based on the existing tool, and we made several significant improvements. First, we create a GUI for the tool to simplify the whole process, as we hope users can breeze through the steps by clicking the 'next' button. Second, we make our tool independent of a specific project, which means users does not need to repeat setting configurations when he wants to use the tool on other projects. We achieve this by integrating the operations into a standalone app. As a result, we can easily use this tool on different platforms and environments. Besides, different from the original tool, which forces users to run the complete process to get the result, the modified tool allows users to choose whether to generate a file in the list. This modification can make our tool more flexible. In addition, we realize some useful features based on the original tool, like database password encryption and database connection testing. The tool can also automatically read business tables from the database, while with the old one, users must input the table names by themselves.

To evaluate our tool, we implemented it and applied it to a web blog application based on Vue, Spring boot, and Mybatis. Using the tool on the database, we can easily connect to the database, configure the parameters easily, choose the tables to generate and finally get our desired files in a single application.

The tool makes the following contributions:

- Improvement on an existing code generation tool.
- An implementation of the tool that is publicly available.
- An evaluation of the tool that shows that it can help

increase efficiency by simplifying tedious steps.

2 RELATED WORKS

In this section, we present some prior work on database-related code generation tools. Andrii et al. [1] propose an approach to automatic source code generation driven by business rules. The approach considers business rules as input, uses the triplestore model for knowledge representation based on business rules, utilizes association rules to suggest attribute data types, and produces an abstract data model. However, obtained verification and validation results show naming issues and generic exception misuse in generated Java Beans. Branko et al. [2] present a design of a tool for automatic code generation for database-oriented web applications, providing fully functional standardized software modules. It can be used efficiently in database-oriented web applications with many standardized pages, and the web interface is the only type of user interface used. However, since the tool concept ties it to application architectures without the application server layer, it is unsuitable for code generation applications that follow the full J2EE architecture.

3 PRELIMINARIES

This section provides some relevant background information and introduces the terminology used in the paper. Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once and run anywhere, meaning that compiled Java code can run on all platforms that support Java without recompiling. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

The Spring Framework is an application framework and inversion of the control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

MyBatis is a Java persistence framework that couples objects with stored procedures or SQL statements using an XML descriptor or annotations. Unlike ORM frameworks, MyBatis does not map Java objects to database tables but Java methods to SQL statements.

Model–view–controller (MVC) is a software architectural pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. Traditionally used for desktop graphical user interfaces, this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate the implementation of the pattern.

JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms. Written as a Java API, JavaFX application code can reference APIs from any Java library. For example, JavaFX applications can use Java API libraries to access native system capabilities and connect to server-based middleware applications.

Apache FreeMarker is a free Java-based template engine, originally focusing on dynamic web page generation with MVC software architecture. However, it is a general-purpose template engine, with no dependency on servlets or HTTP or HTML, and is thus often used for generating source code, configuration files or e-mails.

4 SOLUTIONS

4.1 Scrum for project management

There are lots of software engineering techniques applied for our project such as using Scrum for project management, using requirements elicitation methods to determine and optimize requirements, using Technical Debt to develop more efficient and using code review to improve code quality and decrease bugs, using unit testing and automatic testing. This report will use project management as an example, to illustrate how we use Scrum to manage our project more efficiently.

As we know, our team contains eight members, which is not a small-scale team. What we want to develop is a software engineering tool in which we realize our original new idea. This project is heavy. Therefore, we recognize that it is necessary for us to use a scientific method for project management. After the first discussion, we decide to use scrum to manage our project.

First, according to the scrum framework and the specific situation of our project, we define several roles, the Table 1 shows the members each role contains. Project Manager (PM) is responsible for affairs about project management, such as deciding contents of each iteration, assigning tasks for each role, organizing review meeting for each iteration, establishing project milestones, coordinating the project progress, responding to emergencies etc. Technology Manager (TM) is responsible for affairs about technology management, such as organizing project technical meeting, and code review meeting, choosing project technical frameworks, providing ideas for solving difficult technical problems etc. As for Product Manager (UI designer), the responsibility is to determine and optimzie requirements, complete prototypes, design and optimize UI style of static pages. The last two roles are testers and developers. The develop team has three members, they need to develop various modules and functions according to the requirements, also they need to assure code quality and certain percentage of test coverage. The test team has three members, they need to test and evaluate project by various software testing methods, such Unit Testing, Functional Testing, Regression Testing, Automated Testing and so on. What I want to illustrate is that defining these roles under Scrum principles, is just to clear responsibilities, which does not mean we must strictly obey this division. In actual work, we flexibly adjust the division of labour according to each iteration. After all, cooperation and mutual help is most important.

Role	Members
Project Manager (PM)	Di LIN
Technology Manager (TM)	Di LIN
Product Manager (UI designer)	Jiangwei JIN
Test team	Rui ZHAO, Peiling LIU, Yuewei QIU
Develop team	Lianjun LI, Zhe CHEN, Zhehao JIN

Table 1. members of project roles

After deciding roles and roughly responsibility of each role, we should decide sprints and what should we complete in each sprint. We decide two weeks as one sprint, so we have five sprints. The Table 2 briefly describe what needs to be done for each sprint.

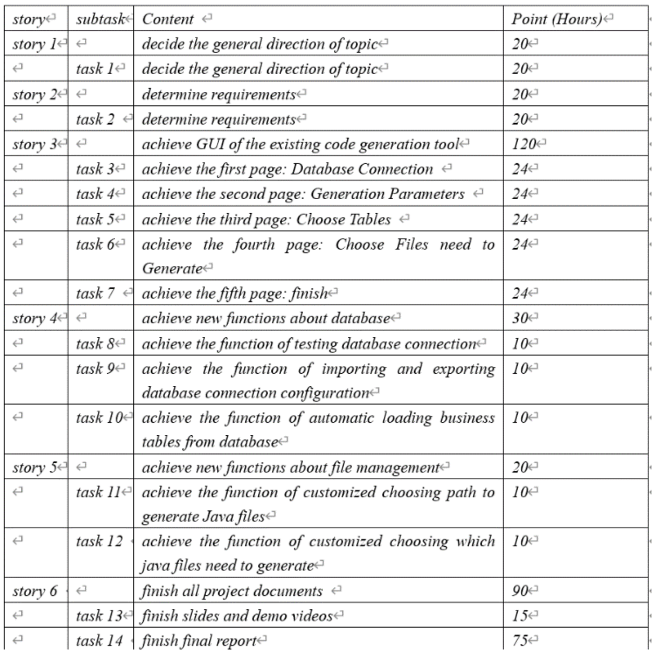
Sprint 1	determine project management approach and the general direction of topic
Sprint 2	search and practice existing tools, determine requirements
Sprint 3	develop and test
Sprint 4	develop and test
Sprint 5	Summarize, finish documents including slides, demo videos and final report

Table 2. contents of each sprint

In the first sprint, we do not rush to work, on the contrary, we first reach an agreement on the approach of project management. As we have mentioned previously, we decide roles, how much sprints we have, what should we do in each sprint, routine meetings for each sprint and so on. After establishing the whole approach of project management, we start to determine the topic. Determine topic is the most important thing, because once the topic is decided it is not proper to change because of the cost and the threat for project progress. Therefore, we establish a grading system to evaluate those topics we initial selected to find the most proper. We evaluate the topic roughly from four dimensions: the practical value for software engineering, the feasibility, the creativity, and the difficulty. In fact, until the first week of the second sprint, we finally determine the improvement of java code generation tool as the topic, because we need to do some attempts earlier to assure the feasibility which spends more time than we expect.

In the second sprint, we search lots of existing projects of GitHub and related research papers. For each existing tool we want to use, we try to learn about it by practice, reading documents even part of source codes, to find the drawbacks and what could improve. After several discussions, we finally choose to base on GreedyStar/Generator (GitHub: https://github.com/GreedyStar/generator/tree/release-1.4.1) and improve because we think this existing tool currently has the highest practical value and most complete functions on the specific domain of Java code generation based on Spring boot & Mybatis templates. Based on that, our improvement on the existing will have enough practical value and creativity. Then we use requirements elicitation methods to decide requirements and create corresponding stories, developing and testing in the third sprint and fourth sprint according to these stories.

After we decide roughly requirements, now we could use various Scrum tools to help project management, which including task board, release burndown chart and sprint backlog. Using requirements elimination methods, we decided three mainly requirements to achieve, each could be a user story contains several different sub tasks. These three stories are achieving GUI of the existing code generation tool, achieving new functions about database, and achieving new functions about file management. Adding the previous two stories which are deciding the general direction of topic and determining requirements, and the last story which is finishing documents, we have six stories in total. Table 3 shows the content and point of each story and its sub tasks.



story	*subtask*	*Content*	*Point (Hours)*
story 1		*decide the general direction of topic*	*20*
	task 1	*decide the general direction of topic*	*20*
story 2		*determine requirements*	*20*
	task 2	*determine requirements*	*20*
story 3		*achieve GUI of the existing code generation tool*	*120*
	task 3	*achieve the first page: Database Connection*	*24*
	task 4	*achieve the second page: Generation Parameters*	*24*
	task 5	*achieve the third page: Choose Tables*	*24*
	task 6	*achieve the fourth page: Choose Files need to Generate*	*24*
	task 7	*achieve the fifth page: finish*	*24*
story 4		*achieve new functions about database*	*30*
	task 8	*achieve the function of testing database connection*	*10*
	task 9	*achieve the function of importing and exporting database connection configuration*	*10*
	task 10	*achieve the function of automatic loading business tables from database*	*10*
story 5		*achieve new functions about file management*	*20*
	task 11	*achieve the function of customized choosing path to generate Java files*	*10*
	task 12	*achieve the function of customized choosing which java files need to generate*	*10*
story 6		*finish all project documents*	*90*
	task 13	*finish slides and demo videos*	*15*
	task 14	*finish final report*	*75*

Table 3. contents and points of these six stories

The third story and sixth story have more points, because we assess that implement GUI and write final report have much more workload than other stories. The total point of the whole project is about 300 points, which means that at least we need 300 hours to finish our project. After establishing all stories and assessing points of each story and its subtasks, now we can draw task board, release burndown chart and sprint backlog.

Story	*Subtask*	*Spring0*	*Sprint1*	*Spring2*	*Sprint3*	*Sprint4*	*Sprint5*
Story 1	*Task 1*	*20*	*0*	*0*	*0*	*0*	*0*
Story 2	*Task 2*	*20*	*20*	*0*	*0*	*0*	*0*
Story 3	*Task 3*	*24*	*24*	*24*	*0*	*0*	*0*
	Task 4	*24*	*24*	*24*	*0*	*0*	*0*
	Task 5	*24*	*24*	*24*	*0*	*0*	*0*
	Task 6	*24*	*24*	*24*	*0*	*0*	*0*
	Task 7	*24*	*24*	*24*	*0*	*0*	*0*
Story 4	*Task 8*	*10*	*10*	*10*	*10*	*0*	*0*
	Task 9	*10*	*10*	*10*	*10*	*0*	*0*
	Task 10	*10*	*10*	*10*	*10*	*0*	*0*
Story 5	*Task 11*	*10*	*10*	*10*	*10*	*0*	*0*
	Task 12	*10*	*10*	*10*	*10*	*0*	*0*
Story 6	*Task 13*	*15*	*15*	*15*	*10*	*10*	*0*
	Task 14	*75*	*75*	*75*	*65*	*50*	*0*

Table 4(a). the sprint backlog at the end of Sprint 2

As you can see in the Table 4(a), we modify the sprint backlog of the lecture because of the situation of our project. Because all the members of our group selected about four to six courses, which include lots of midterm exams, projects, and homework, so we could not spend lots of time on this project which means the dimension of the sprint backlog of the lecture is not really proper for our project. Thus, we modify the dimension of the sprint backlog, to check the progress of stories at the end of each sprint, rather than at the end of each day.

Table 4(a) shows the sprint backlog we have record at the end of the second sprint. At that time, we have finished the first task and second task, the contents of these two stories could be found in the Table 3. There were still four stories not completed at that time, so we planned to finish the third story (implement GUI of the existing tool) and little part of the sixth story at the end of the third sprint, because it seems feasible to start making slides and writing the final report at the same time.

Story	*Subtask*	*Spring0*	*Sprint1*	*Spring2*	*Sprint3*	*Sprint4*	*Sprint5*
Story 1	*Task 1*	*20*	*0*	*0*	*0*	*0*	*0*
Story 2	*Task 2*	*20*	*20*	*0*	*0*	*0*	*0*
Story 3	*Task 3*	*24*	*24*	*24*	*0*	*0*	*0*
	Task 4	*24*	*24*	*24*	*0*	*0*	*0*
	Task 5	*24*	*24*	*24*	*24*	*0*	*0*
	Task 6	*24*	*24*	*24*	*24*	*0*	*0*
	Task 7	*24*	*24*	*24*	*24*	*0*	*0*
Story 4	*Task 8*	*10*	*10*	*10*	*10*	*0*	*0*
	Task 9	*10*	*10*	*10*	*10*	*0*	*0*
	Task 10	*10*	*10*	*10*	*10*	*0*	*0*
Story 5	*Task 11*	*10*	*10*	*10*	*10*	*0*	*0*
	Task 12	*10*	*10*	*10*	*10*	*0*	*0*
Story 6	*Task 13*	*15*	*15*	*15*	*15*	*15*	*0*
	Task 14	*75*	*75*	*75*	*75*	*75*	*0*

Table 4(b). the sprint backlog at the end of Sprint 3

However, the progress of stories is not expected as we want. Time came to the end of the third sprint, we remake the sprint backlog according to the situation of the third sprint, which is shown in the Table 4(b). We found that we merely finished 40% of the third story, which was a great gap. At the meeting, we concluded some situations we did not consider at the end of the second sprint, such as the hesitation of which UI style we should choose, change of requirements, refactor the entire process when halfway developed, all these situations spend lots of extra time. This is the reason we did not finish the third story at the end of the third sprint. Also, we found that it is difficult to make slides and write final report before development completed. To solve the problem the sprint backlog has reflected, we soonly made some measures at the end of the third sprint. This is the example how we use scrums tools, such as the sprint backlog to track the progress of each story, to discover and solve the problem as soon as possible. Without the sprint backlog, maybe we could not detect problems in time, which may further lead to the mess of project management.

4.2 Program Structure

As we have mentioned in the previous parts, our code generation tool is based on an existing open-source code generation project and implement GUI and some new functions related database and file manage, which makes our code generation tool easy and powerful. Therefore, to develop our tool with the most proper and efficient method, we need to understand how java code is auto generated in the existing tool, in other words, we need to get the whole method invoke chain of code generation function, and then we can design program structure of our tool based on it and develop by the program structure.

4.2.1 Program Structure of the existing tool

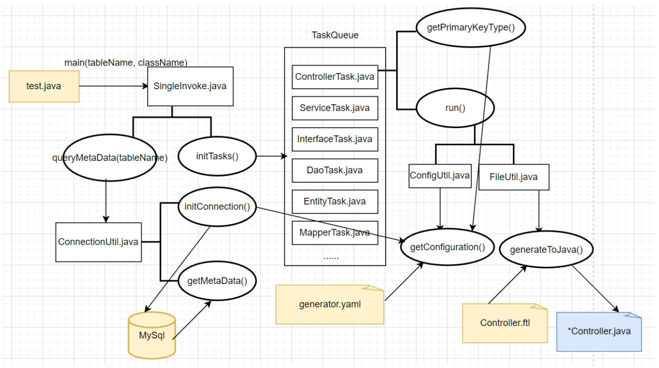


Figure 1. the program structure of the existing tool

We use the generation process of Controller Java file (*Controller.java) of the demo blog project (Using Spring and Mybatis frameworks) to illustrate how the existing tool generate java codes. As the Figure 1 shows, after finishing installing all code generation dependencies and setting the figuration file (generator.yaml) and Java file templates (Contorller.ftl in this example), users could write the test class (test.java), choose business tables and generate corresponding java files. Firstly, The main() method will construct an instance of SingleInvoke class, which will first call queryMetaData() method to initial database connection of the blog project, and load table metadata from database for code generation. Besides, the instance will also call initTask() to initial the task and insert the task into the task queue waiting for executing. When executing the task, for example, the controller task (ControllerTask.java), the task will then mainly call two methods, the first is to load all parameters it need, including database configuration, file name configuration, file path configuration etc. from the configuration file (generator.yaml). After completely loading all parameters need, the task will call generateToJava() method, which is to dynamically load the Java template (Controller.ftl) to parse and set parameters, finally generate codes we want, which is *Controller.java.

4.2.2 Program Structure of our code generation tool

After listing the process of code generation in the existing tool, now we could combine it and our improve points, to design program structure of our tool. The first also the most important improve is to develop GUI for the existing tool, which need us integrate all setting works and other works into GUI. Therefore, we need to change all inputs of the modules and all related interfaces to adapt new inputs from GUI.

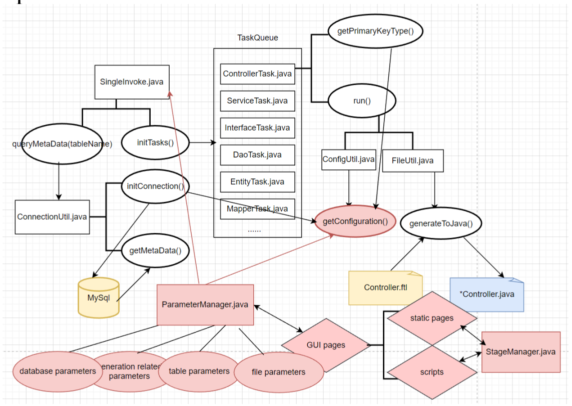


Figure 2. program structure (implement GUI)

As the process shows in Figure 1, there four inputs in the code generation process that have been marked yellow in the figure, which are test.java, MySql database, generator.yaml and controller.ftl. The instance of SingleInvoke class previously load business table names and class names directly from test class, which should change to load directly from GUI pages. The existing tool load database parameters and generation related parameters (file name, file path etc.) from the configuration file (generator.yaml) by method getConfiguration(), now should change to load configurations from GUI pages. To reduce develop cost and reuse codes, we do not need develop a new method to replace getConfiguration(), we only need to change interfaces and logics of getConfiguration() so that it could be compatible well with GUI pages. Furthermore, we also need a parameter manager block to manage and cache parameters and configurations of each page, and a page manager block to manage and cache all static pages and scripts, as the Figure 2 shows.

To develop new file related functions and database related functions, we need to extend the program structure of our code generation tool. Because we add functions such as automatic load tables from database, customized choose which files to generate, import and export database connection configuration and customized choose folder to generate codes, we need to manage new configurations and parameters in the parameter manage block, and new pages in the page manage block. Based on the structure which has implement GUI, we could futher implement those new functions.

4.3 develop process

4.3.1 chose the proper GUI framework (JavaFX versus Swing)

Swing and JavaFX are the two main JAVA-based GUI development toolkits available today. Both technologies are used to build Graphical User Interfaces (GUI) for Java-based applications. JavaFX and Swing both support "skinning", which is the process of changing the visual appearance of a GUI by applying a skin [3]. We had a framework selection discussion before the GUI design, and at first, both swing and JavaFX were taken into consideration. To go for a more appropriate model, we compared the two.

Swing was the main driver for WYSIWYG (What You See Is What You Get) editors. These editors were mostly generating Java code from the visual representation, a code that was hard to modify and maintain. However, as an alternative, JavaFX comes with Scene Builder and FXML, which enable the controller does not need to look up the UI elements it needs to interact with and the invocation of its event listener methods is handled by the FXML logic [4].In that case, JavaFX gives developers a chance to design the interface much faster, without writing and maintaining a lot of boilerplate code, and focusing on the application logic. So, we finally adopt JavaFX for use as it seems that JavaFX might be a better choice for our project.

4.3.2 design UI and static pages by Scene Builder

When we use JavaFX to write a GUI, we often feel that doing the interface design in code is a very troublesome task. However, javaFX has a built-in UI builder for the visual interface, the scene builder, which can automatically generate FXML files by simply clicking your mouse over the interface without requiring you to manually write code.

At the center of this approach of UI design is the FXML file. We could see the process of how FXML works in Figure 3. The FXML can declare an associated controller class and expose to it UI elements, and event handler hooks. The controller is then responsible for reacting to the events and updating the view accordingly. It is an XML file format designed specifically to hold information about UI elements. It contains the "what" of the UI elements, but not the "how." [4] At its core, FXML is a Java object serialization format that can be used for any Java classes written in a certain way, including all old-style JavaBeans.

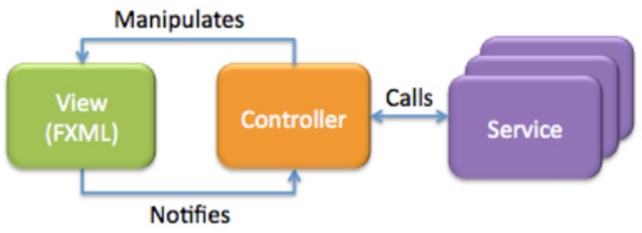


Figure 3. the working process of FXML

Therefore, we use the scene builder to generate static pages (.yaml format files) and complete the UI design by dragging and dropping through the interface with visual tools. The system will generate FXML files and use the controller class to control all the actions inside the interface, which can make the interface separate from the code. Figure 4 is the design view of a button we designed in our project using the scene builder and the FXML it automatically generates.

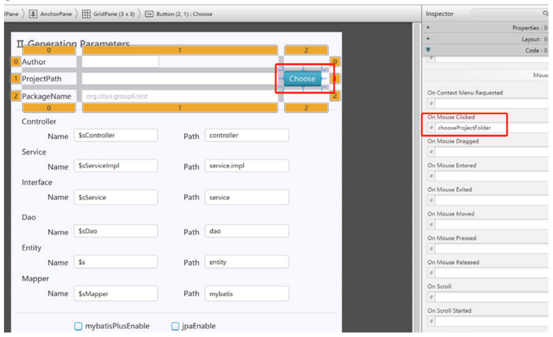
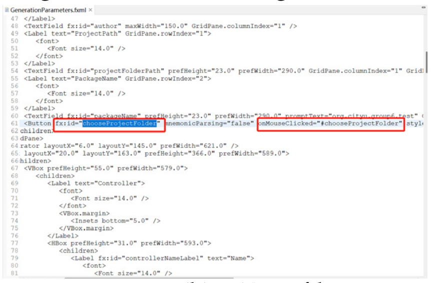


Figure 4(a). Visual design interface

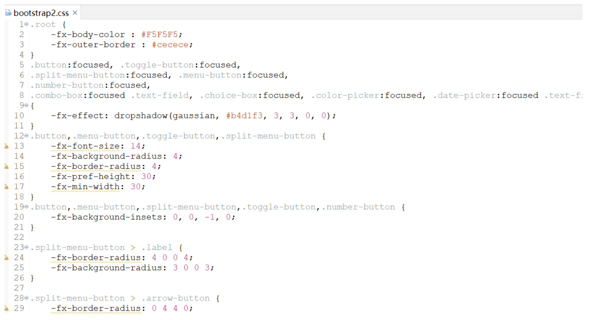


GenerationParameters.fxml

```
</Label>
<TextField fx:id="author" maxWidth="150.0" GridPane.columnIndex="1" />
<Label text="ProjectPath" GridPane.rowIndex="1">
   <font>
      <Font size="14.0" />
   </font>
</Label>
<TextField fx:id="projectFolderPath" prefHeight="23.0" prefWidth="290.0" GridPane.columnIndex="1" Grid
<Label text="PackageName" GridPane.rowIndex="2">
   <font>
      <Font size="14.0" />
   </font>
</Label>
<TextField fx:id="packageName" prefHeight="23.0" prefWidth="290.0" promptText="org.cityu.group6.test" 
<Button fx:id="chooseProjectFolder" mnemonicParsing="false" onMouseClicked="#chooseProjectFolder" styl
children>
dPane>
rator layoutX="6.0" layoutY="145.0" prefWidth="621.0" />
 layoutX="20.0" layoutY="163.0" prefHeight="366.0" prefWidth="589.0">
hildren>
<VBox prefHeight="55.0" prefWidth="579.0">
   <children>
      <Label text="Controller">
         <font>
            <Font size="14.0" />
         </font>
         <VBox.margin>
            <Insets bottom="5.0" />
         </VBox.margin>
      </Label>
      <HBox prefHeight="31.0" prefWidth="593.0">
         <children>
            <Label fx:id="controllerNameLabel" text="Name">
               <font>
                  <Font size="14.0" />
```

Figure 4(b). FXML file

Therefore, we can use Scene Builder to design all five user interfaces, including all events binding on the page. Furthermore, we use CSS files to optimize the UI style, also, make the UI style of all user interfaces consistent, as the Figure 5 shows below.



bootstrap2.css

```
.root {
    -fx-body-color : #F5F5F5;
    -fx-outer-border : #cecece;
}
.button:focused, .toggle-button:focused,
.split-menu-button:focused, .menu-button:focused,
.number-button:focused,
.combo-box:focused .text-field, .choice-box:focused, .color-picker:focused, .date-picker:focused .text-f
{
    -fx-effect: dropshadow(gaussian, #b4d1f3, 3, 3, 0, 0);
}
.button,.menu-button,.toggle-button,.split-menu-button {
    -fx-font-size: 14;
    -fx-background-radius: 4;
    -fx-border-radius: 4;
    -fx-pref-height: 30;
    -fx-min-width: 30;
}
.button,.menu-button,.split-menu-button,.toggle-button,.number-button {
    -fx-background-insets: 0, 0, -1, 0;
}

.split-menu-button > .label {
    -fx-border-radius: 4 0 0 4;
    -fx-background-radius: 3 0 0 3;
}

.split-menu-button > .arrow-button {
    -fx-border-radius: 0 4 4 0;
```

Figure 5(a). CSS file (bootstrap)

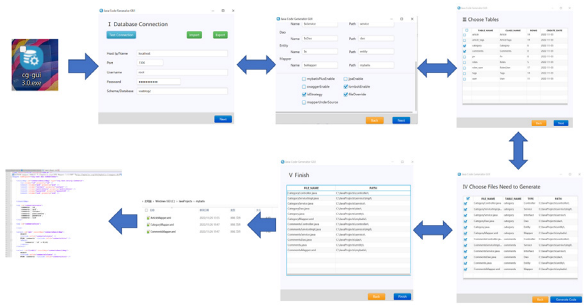
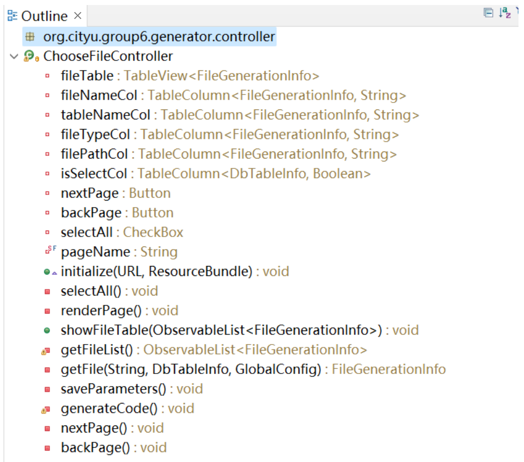


Figure 5(b). the whole process and all pages

4.3.3 Example of the fourth page (Choose Files Need to Generate)

After UI deign and static pages generated by Scene Builder, now we need to develop controller java files and service java files to control the process of GUI, listen and respond to the events and implement new functions. We use the fourth page of our coded generation tool which named "Ⅳ Choose Files Need to Generate" to illustrate how we develop a GUI page and related functions, because this page is the most important page achieve two functions: customized choose generated files based on the business tables selected on last page and generate those selected files.



```
Outline
org.cityu.group6.generator.controller
ChooseFileController
    fileTable : TableView<FileGenerationInfo>
    fileNameCol : TableColumn<FileGenerationInfo, String>
    tableNameCol : TableColumn<FileGenerationInfo, String>
    fileTypeCol : TableColumn<FileGenerationInfo, String>
    filePathCol : TableColumn<FileGenerationInfo, String>
    isSelectCol : TableColumn<DbTableInfo, Boolean>
    nextPage : Button
    backPage : Button
    selectAll : CheckBox
    pageName : String
    initialize(URL, ResourceBundle) : void
    selectAll() : void
    renderPage() : void
    showFileTable(ObservableList<FileGenerationInfo>) : void
    getFileList() : ObservableList<FileGenerationInfo>
    getFile(String, DbTableInfo, GlobalConfig) : FileGenerationInfo
    saveParameters() : void
    generateCode() : void
    nextPage() : void
    backPage() : void
```

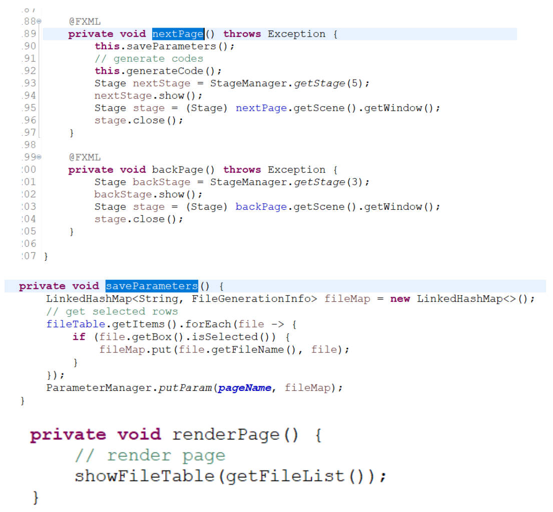
Figure 6. the outline of ChooseFileController.java

ChooseFileController.java is the java file which responsible for listening and responding all the events bound on the static page (ChooseFileController.fxml). Figure 6 shows the outline of ChooseFileController.java, we can find that this java file roughly consists of three parts, the first part are those elements on the static page, including TableView, TableColumn, Button, CheckBox. We use the FXML annotation (@FXML) map these fields to the corresponding elements on the static page (ChooseFileController.fxml), as the Figure 7 shows below.

```
@FXML
private TableView<FileGenerationInfo> fileTable;
@FXML
private TableColumn<FileGenerationInfo, String> fileNameCol, tableNameCol, fileTypeCol, filePathCol;
@FXML
private TableColumn<DbTableInfo, Boolean> isSelectCol;
@FXML
private Button nextPage;
@FXML
private Button backPage;
@FXML
private CheckBox selectAll;
```

Figure 7. using @FXML to map fields and FXML elements

The second part are those common methods nearly in every page in our code generation tool, which could be abstracted into interfaces. There methods are initialize(), renderPage(),saveParameters(), generateCode(), nextPage() and backPage() as Figure 8 shows below. The nextPage() method and backPage() method implement all logics about turning to last page or next page, including save parameters and configurations of this page to the parameter cache block by calling saveParameters(), open next page or last page before close current page and so on. The method initialize() responsible for executing those things need to do when initializing the page, in this page, is to render the table which contains java files for users to choose based on the database tables selected on the third page.



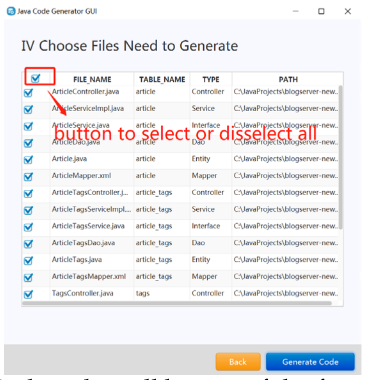
```java
    @FXML
    private void nextPage() throws Exception {
        this.saveParameters();
        // generate codes
        this.generateCode();
        Stage nextStage = StageManager.getStage(5);
        nextStage.show();
        Stage stage = (Stage) nextPage.getScene().getWindow();
        stage.close();
    }

    @FXML
    private void backPage() throws Exception {
        Stage backStage = StageManager.getStage(3);
        backStage.show();
        Stage stage = (Stage) backPage.getScene().getWindow();
        stage.close();
    }

}
```

```java
  private void saveParameters() {
      LinkedHashMap<String, FileGenerationInfo> fileMap = new LinkedHashMap<>();
      // get selected rows
      fileTable.getItems().forEach(file -> {
          if (file.getBox().isSelected()) {
              fileMap.put(file.getFileName(), file);
          }
      });
      ParameterManager.putParam(pageName, fileMap);
  }
```

```java
private void renderPage() {
    // render page
    showFileTable(getFileList());
}
```

Figure 8. common methods of the page

The last part are methods about events and data of the file table. The method selectAll() is used for listening and responding click events of the select-all button, which is shown in Figure 9.



Java Code Generator GUI

IV Choose Files Need to Generate

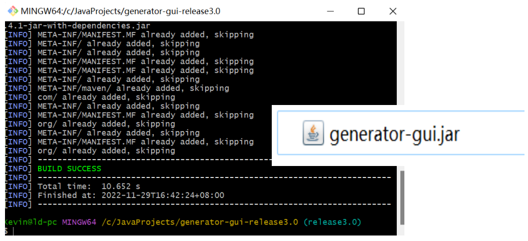
✓	FILE_NAME	TABLE_NAME	TYPE	PATH
✓	ArticleController.java	article	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleServiceImpl.java	article	Service	C:\JavaProjects\blogserver-new...
✓	ArticleService.java	article	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleDao.java	article	Dao	C:\JavaProjects\blogserver-new...
✓	Article.java	article	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleMapper.xml	article	Mapper	C:\JavaProjects\blogserver-new...
✓	ArticleTagsController.j...	article_tags	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleTagsServiceImpl....	article_tags	Service	C:\JavaProjects\blogserver-new...
✓	ArticleTagsService.java	article_tags	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleTagsDao.java	article_tags	Dao	C:\JavaProjects\blogserver-new...
✓	ArticleTags.java	article_tags	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleTagsMapper.xml	article_tags	Mapper	C:\JavaProjects\blogserver-new...
✓	TagsController.java	tags	Controller	C:\JavaProjects\blogserver-new...

Figure 9. the select-all button of the fourth page

The method getFileList() and getFile() is to generate file list used for render the file table. Those file names and file paths are generated by the configurations which set in the second page, using getParam() method of ParameterManager class to get those parameters from the cache block. And the last method showFileTable() is to render the file table of the static page, called by renderPage() when first enter to this page.

4.4 Package to executable file

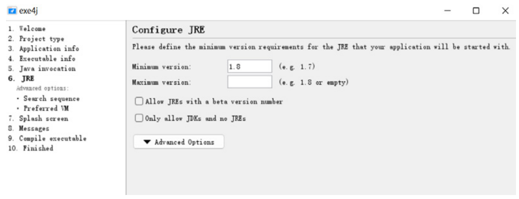
To further let users to use our java code generation tool easier, we decided to package our code generation tool to an executable file (.exe format) by two steps. The first step is to use maven (command: mvn pakcage) to package our code generation tool to an executable jar format file with all dependencies, the second step is to use exe4j (a Java exe maker that helps integrate Java applications into the Windows operating environment) to package our jar file into the executable file (.exe format). This executable file can directly run in the Windows Environment (32 bit/ 64 bit) with JRE (Java Runtime Enviroment) which further make this tool more convenient. The excutable file also included in our source code. We want to compatible with Linux and macOS in the future.



MINGW64:/c/JavaProjects/generator-gui-release3.0

```
.4.1-jar-with-dependencies.jar
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/maven/ already added, skipping
[INFO] com/ already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] org/ already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] org/ already added, skipping
[INFO] ------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.652 s
[INFO] Finished at: 2022-11-29T16:42:24+08:00
[INFO] ------------------------------------------------------------------------

kevin@ld-pc MINGW64 /c/JavaProjects/generator-gui-release3.0 (release3.0)
$
```

Figure 10. use maven to package into jar file



exe4j

1. Welcome
2. Project type
3. Application info
4. Executable info
5. Java invocation
6. JRE
   Advanced options:
   - Search sequence
   - Preferred VM
7. Splash screen
8. Messages
9. Compile executable
10. Finished

Configure JRE

Please define the minimum version requirements for the JRE that your application will be started with.

Minimum version: 1.8 (e.g. 1.7)
Maximum version: (e.g. 1.8 or empty)

☐ Allow JREs with a beta version number
☐ Only allow JDKs and no JREs

Figure 11. use exe4j to pakcage into .exe file

5 EVALUATIONS

5.1 Experiment setup

In this section, we want to know if our program is working correctly and efficiently, and what could be improved. We will address three areas: validation, comparison, and limitation. We will use tools and experimentation to test the functionality of our program to ensure that it does the job of code generation correctly. In addition to ensuring that the code is generated correctly, we also need to ensure that our program is effective, meaning that it is simpler, easier, and faster to use than the original program, i.e., the ease of use that we mentioned earlier.

5.2 Validation

In this section, we want to know if the various functions of our program are working correctly. Can we successfully connect to the database? Is it fully responsive to our individual choices? Can we successfully import the data tables in it, write them to the program and generate codes automatically? Does it follow our options correctly? The following Figure 12 shows each functional module in the execution flow that we will test, and we will explore the success of the calls between each module.

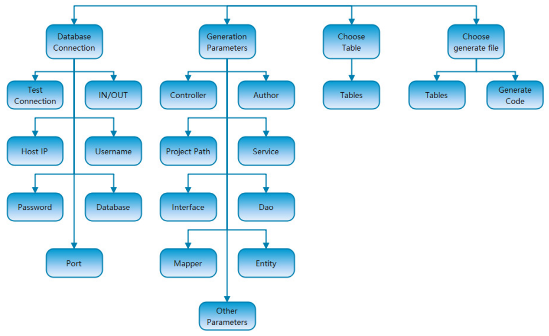


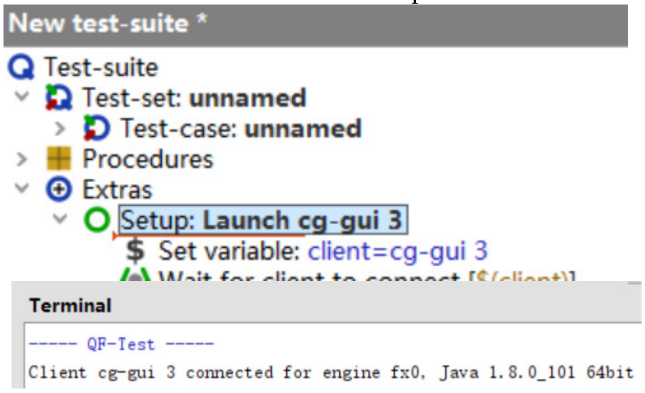
Figure12. functional module

5.2.1 Test method

QF-Test from Quality First Software is a cross-platform software tool for GUI test automation specialized on Java [5]. Simple and efficient, QF-Test captures test steps and GUI components through recording and executes tests through playback, which displays the test process in a tree node pattern [6].
In order to conduct a large number of tests quickly and in a short period of time, we used QF, an automated testing tool, to help us eliminate the tedious process of inputting content and selecting options each time manually and conduct fully automated machine testing.

5.2.2 Test process

Launch QF-test, find the .exe file we need to test by selecting the folder, import the file into the test list, click the "run" button directly to start the test and click the "recording" button to record the test process [7]. Manually enter the Host IP, Port number, Username, Database password, and Database, and click next. (When testing import/export, import the .txt format settings file directly through the external folder instead of entering them manually). After that, the system will automatically repeat the automatic test several times according to the previously manually set parameters and automatically generate the test file after the test is completed.


New test-suite *
Test-suite
Test-set: unnamed
Test-case: unnamed
Procedures
Extras
Setup: Launch cg-gui 3
Set variable: client=cg-gui 3

Terminal

```
----- QF-Test -----
Client cg-gui 3 connected for engine fx0, Java 1.8.0_101 64bit
```

Figure 13. Launch successfully

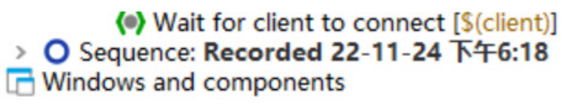
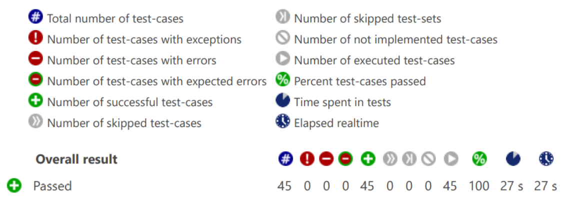

Wait for client to connect [$(client)]
Sequence: Recorded 22-11-24 下午6:18
Windows and components

Figure 14. Test file

5.2.3 Test result

After 45 rounds of testing using QF-test, we checked the automatically generated test files and so far, no problems were found in the program from the automatic tests, which can indicate that the code generation tool we designed is available and working correctly.



Total number of test-cases	Number of skipped test-sets
Number of test-cases with exceptions	Number of not implemented test-cases
Number of test-cases with errors	Number of executed test-cases
Number of test-cases with expected errors	Percent test-cases passed
Number of successful test-cases	Time spent in tests
Number of skipped test-cases	Elapsed realtime

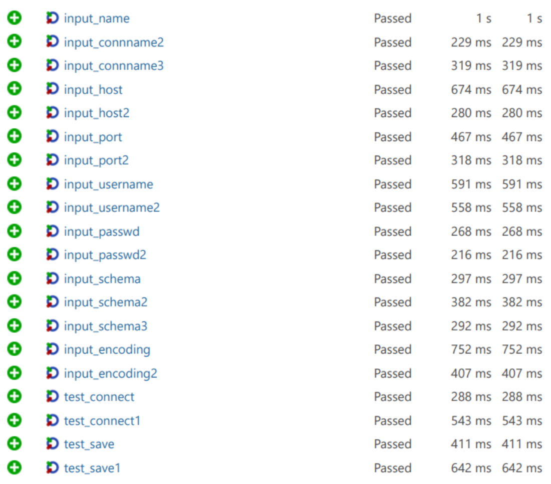
Overall result												
Passed	45	0	0	0	45	0	0	0	45	100	27 s	27 s

Figure 15. Overall Overview Test Report

UI display: Double click on the .exe file, the program can successfully open and display the UI interface. The modules in each interface are displayed normally.

UI_test	Passed	27 s	27 s

Database connection page: Here we will enter the basic information including Host IP, Port number, Username, Database password, and Database name, we need to connect to the database which contains the tables we need to use to generate code. Here we also verify that importing the established parameter configuration directly from outside and exporting the parameter configuration entered on this page to the set path for storage works correctly. It is related to the transfer and import of information flow, we tested whether the program successfully entered the information we entered from the text box of the UI interface. The table shown below illustrates that the database connection page runs successfully and correctly.



input_name	Passed	1 s	1 s
input_connname2	Passed	229 ms	229 ms
input_connname3	Passed	319 ms	319 ms
input_host	Passed	674 ms	674 ms
input_host2	Passed	280 ms	280 ms
input_port	Passed	467 ms	467 ms
input_port2	Passed	318 ms	318 ms
input_username	Passed	591 ms	591 ms
input_username2	Passed	558 ms	558 ms
input_passwd	Passed	268 ms	268 ms
input_passwd2	Passed	216 ms	216 ms
input_schema	Passed	297 ms	297 ms
input_schema2	Passed	382 ms	382 ms
input_schema3	Passed	292 ms	292 ms
input_encoding	Passed	752 ms	752 ms
input_encoding2	Passed	407 ms	407 ms
test_connect	Passed	288 ms	288 ms
test_connect1	Passed	543 ms	543 ms
test_save	Passed	411 ms	411 ms
test_save1	Passed	642 ms	642 ms

Choose table page: On this page, the database connection is successful based on the database connection interface, and all available tables in that database are displayed on the page, so you can see that the data flow was transferred successfully.
Choose file generate page: Like the previous page, on this page you can obviously see that the files selected on the previous page are shown in the table on this page, which means that the data transfer was successful. The difference with the previous page is that there is a generate code button to select the file for automatic code generation. We cannot visually conclude whether the code generation is successful or not, but this is reflected in the test file.

generate_code	Passed	3 s	3 s

5.3 Comparison

5.3.1 Enhance Functions

The existing code generation tool is a good tool for generating Java code based on database. The code template uses the current mainstream Java framework: Spring, SpringMVC, Mybatis. It is useful for our developers. However, when we use the tool, we found that the there are some inconveniences. Some functions of the tool are not very convenient. There are some certain limitations of using functions. So, in addition to add GUI for the tool, we also enhanced the tool's some functions and add more functions that we think it is useful. For example, we add file related functions, some checking functions and add the ability to save and invoke settings functions. After enhancing and adding these functions, the tool become more user-friendly and powerful for developers.

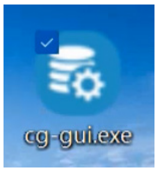


Figure 16. the executable file of our code generation tool

5.3.2 File function

In the existing code generation tool, the code has specified the generation path. In the case of SpringMVC, which is a famous Java framework, the tool will generate the files fixed at the 'src\main\java' path. Under this fixed path, the tool will generate the corresponding the files and folders based on the relevant information. Here is an example of path setting:
path:
  controller: controller
  service: service.impl
  dao: org.com.blog.dao
According to the parameters of path, the tool will generate such files and folders below:

```
controller/XXXController.java
service/impl/XXXService.java
dao/org/com/blog/XXXDao.java
```

As we can see the rules of generating the files is that the tool breaks down the input contents from left to right, using dots as partition markers. Then the tool creates folders in the same order as the input and generate all the files you need in the last folder. For example, 'service: service.impl' means that all 'XXXService.java' files will be generated under the path of 'service\impl\'. Similarly, all 'XXXDao.java' files will be generated under the path of 'org\com\blog\dao'. But note that these paths have a big prefix path, which is 'src\main\java\'. What I have to say is that all 'XXXDao.Java' files actually generated in 'src\main\Java\org\com\blog\dao', this file path. If the developer wants all dao files to be generated under the org\com\main path, this tool does not directly meet the needs of the developer. The developer needs to move all the files to the file path he wants to generate after the files are generated.
Therefore, the original tool has some limitations for this operation of generating files. It generates files in the same path as the tool, which means that the developer often needs to move the generated files to the path he wants. Combined with the design of GUI for the original tool, we add the ability to select a path so that developers can directly generate files below their desired file path.
As the developer chooses the file path, the new tool can directly generate all the files under the selected file path.

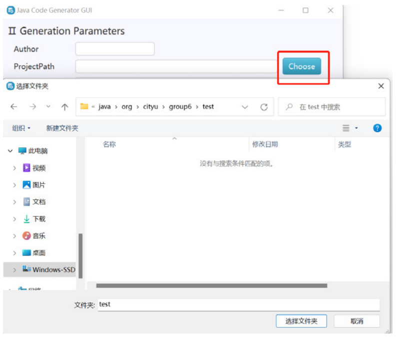
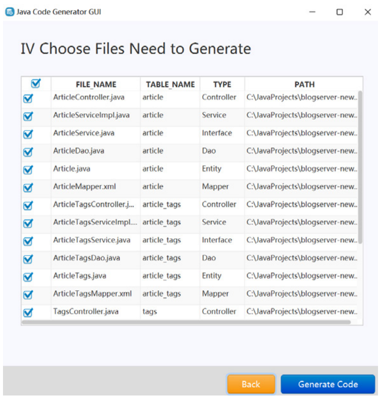


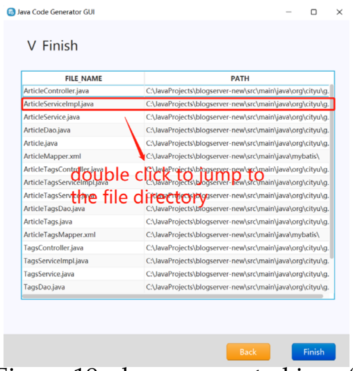
Figure 17. select file path



IV Choose Files Need to Generate

✓	FILE_NAME	TABLE_NAME	TYPE	PATH
✓	ArticleController.java	article	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleServiceImpl.java	article	Service	C:\JavaProjects\blogserver-new...
✓	ArticleService.java	article	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleDao.java	article	Dao	C:\JavaProjects\blogserver-new...
✓	Article.java	article	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleMapper.xml	article	Mapper	C:\JavaProjects\blogserver-new...
✓	ArticleTagsController.j...	article_tags	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleTagsServiceImpl....	article_tags	Service	C:\JavaProjects\blogserver-new...
✓	ArticleTagsService.java	article_tags	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleTagsDao.java	article_tags	Dao	C:\JavaProjects\blogserver-new...
✓	ArticleTags.java	article_tags	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleTagsMapper.xml	article_tags	Mapper	C:\JavaProjects\blogserver-new...
✓	TagsController.java	tags	Controller	C:\JavaProjects\blogserver-new...

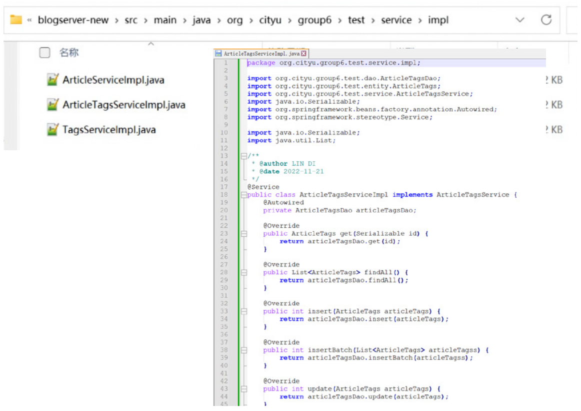
Figure 18. select files to generate



V Finish

FILE_NAME	PATH
ArticleController.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleServiceImpl.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleService.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleDao.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
Article.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleMapper.xml	C:\JavaProjects\blogserver-new\src\main\java\mybatis\
ArticleTagsController.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleTagsServiceImpl.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleTagsService.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleTagsDao.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleTags.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
ArticleTagsMapper.xml	C:\JavaProjects\blogserver-new\src\main\java\mybatis\
TagsController.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
TagsServiceImpl.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
TagsService.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.
TagsDao.java	C:\JavaProjects\blogserver-new\src\main\java\org\cityu\g.

Figure 19. show generated java files


blogserver-new › src › main › java › org › cityu › group6 › test › service › impl
名称
ArticleServiceImpl.java
ArticleTagsServiceImpl.java
TagsServiceImpl.java
ArticleTagsServiceImpl.java

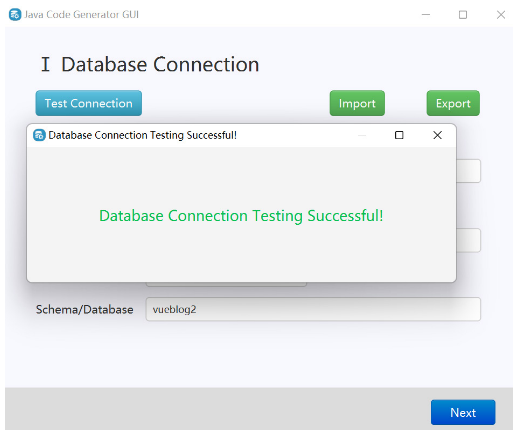
```java
package org.cityu.group6.test.service.impl;

import org.cityu.group6.test.dao.ArticleTagsDao;
import org.cityu.group6.test.entity.ArticleTags;
import org.cityu.group6.test.service.ArticleTagsService;
import java.io.Serializable;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.io.Serializable;
import java.util.List;

/**
 * @author LIN DI
 * @date 2022-11-21
 */
@Service
public class ArticleTagsServiceImpl implements ArticleTagsService {
    @Autowired
    private ArticleTagsDao articleTagsDao;

    @Override
    public ArticleTags get(Serializable id) {
        return articleTagsDao.get(id);
    }

    @Override
    public List<ArticleTags> findAll() {
        return articleTagsDao.findAll();
    }

    @Override
    public int insert(ArticleTags articleTags) {
        return articleTagsDao.insert(articleTags);
    }

    @Override
    public int insertBatch(List<ArticleTags> articleTagss) {
        return articleTagsDao.insertBatch(articleTagss);
    }

    @Override
    public int update(ArticleTags articleTags) {
        return articleTagsDao.update(articleTags);
    }
```

Figure 20. check generated java codes

In the existing tool, the projectFolderPath is a variable used to store the file path chosen by the developer. The addition of the packageName variable is to take into account the development habits of many developers. Developers will express the development project in the form of multiple layers of files, for example, 'org.com.blog' means that the project files are stored in the '\org\com\blog' file path. If the developer chooses 'D:\workspace\' as the file path, which means the variable of projectFolderPath is 'D:\workspace\' and the variable of packageName is 'org.com.blog', the result is that the all files will be stored in 'D:\workspace\org\com\blog'. packageName also has a default value of 'scr.main.java'. Of course, if projectFolderPath is 'D:\workspace\org\com\blog' and packageName will still be 'org.com.blog', which means that the second half of the file path coincides with packageName, the new tool will not create the file in a rigid way. Instead, the file is generated directly in the corresponding path. In other words, there is no such file path like 'D:\workspace\org\com\blog\org\com\blog'. Let's take the SpringMVC Java framework as an example again. The Controller, Service and Dao files can be generated like the example above in the new function. We just improved the file function so that the operation of generating code can be more freedom and convenience. And the developer does not need to create the java project in IDE in advance and then use the tool to generate code. He can directly generate the codes he needs in the new tool by using the new function.

5.3.3 Detection function

We separate the process of developers inputting database information from the main program and design a special window for developers to input database information and provide detection functions. After the developer has entered the database information, he can click the Test Connection button to check if the database connection was successful. When the database connection is successful, a popup will be displayed. Otherwise, the tool will not respond. We think is good way to solve the problem that the tool fails to generate code. The developer can quickly react to database information entered incorrectly and correct it when a database connection check fails.



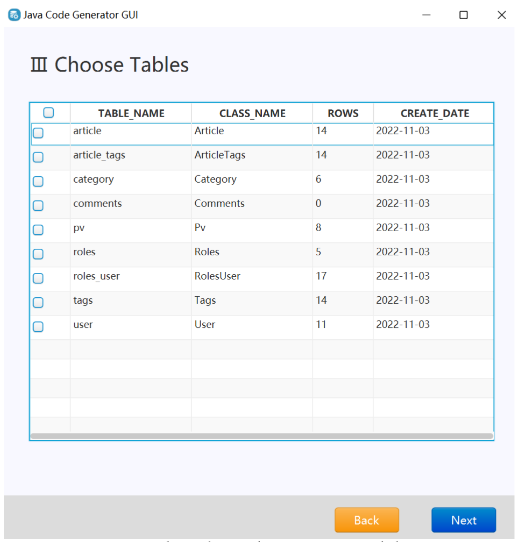
Database Connection Testing Successful!

Database Connection Testing Successful!

Schema/Database: vueblog2

Figure 21. test connection

In addition to the data connection detection, we also developed several additional steps of detection and validation, which are to confirm the tables in the database, confirm all files that will be generated, and confirm the generated files, while in the ogriginal tool, we must write extra code to do these.



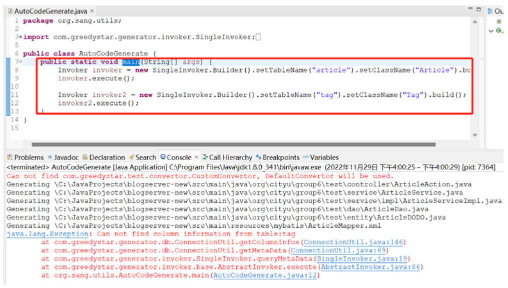
	TABLE_NAME	CLASS_NAME	ROWS	CREATE_DATE
☐	article	Article	14	2022-11-03
☐	article_tags	ArticleTags	14	2022-11-03
☐	category	Category	6	2022-11-03
☐	comments	Comments	0	2022-11-03
☐	pv	Pv	8	2022-11-03
☐	roles	Roles	5	2022-11-03
☐	roles_user	RolesUser	17	2022-11-03
☐	tags	Tags	14	2022-11-03
☐	user	User	11	2022-11-03

Figure 22. loading business tables

For the function of confirming the tables in the database, the tool will display all the tables in the connected database in a certain format in the third step, which is the step after Generation Parameters step. Developers can confirm the part parameters of the table of the database. It's also a way for developers to make sure they're connecting to the database they want. However, this step is more about selecting functions than confirming them. The selection function will be explained in detail in a later section.
Confirming the files that will be generated is also mainly used in section function. The tool will generate all files based on the selected table. Developers can confirm and make certain choices.
Finally, after the developer has made all the choices, the tool will package all the generated files and their related information.
In the fifth step, the tool will show each successfully generated file and the path to the generated file. The developer can verify that the tool has successfully generated the file by double-clicking on the file he wants to confirm and opening its path.
Detection function is, in other words, adding multiple breakpoints to the code. It gives the developer more Windows to check that the program is working the way he expects it to, and it allows him to stop the program from running if it deviates. This change can reduce the loss caused by the error of the operation to some extent. We believe that this makes the tool more flexible.

5.3.4 Selecting function

In the original tool, it also provides options for developers. The developer can choose which database tables to generate, as well as which files to generate. But in the actual use process, we found that the selection function of the original tool has space for improvement. For example, when choosing which database files to generate, developers need to create their own 'AutoCodeGenerate.java' file in the 'utiles' file and enter the following code in it:



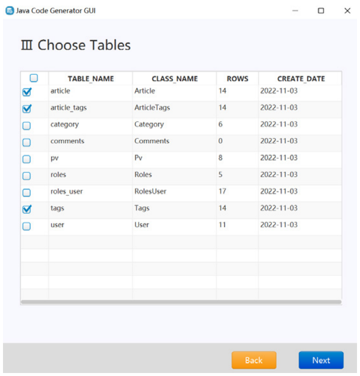
AutoCodeGenerate.java

```
package org.sang.utils;

import com.greedystar.generator.invoker.SingleInvoker;

public class AutoCodeGenerate {
    public static void main(String[] args) {
        Invoker invoker = new SingleInvoker.Builder().setTableName("article").setClassName("Article").bu
        invoker.execute();

        Invoker invoker2 = new SingleInvoker.Builder().setTableName("tag").setClassName("Tag").build();
        invoker2.execute();
    }
}
```

```
<terminated> AutoCodeGenerate [Java Application] C:\Program Files\Java\jdk1.8.0_341\bin\javaw.exe (2022年11月29日 下午4:00:25 – 下午4:00:29) [pid: 7364]
Can not find com.greedystar.test.convertor.CustomConvertor, DefaultConvertor will be used.
Generating \C:\JavaProjects\blogserver-new\src\main\java\org\cityu\group6\test\controller\ArticleAction.java
Generating \C:\JavaProjects\blogserver-new\src\main\java\org\cityu\group6\test\service\ArticleService.java
Generating \C:\JavaProjects\blogserver-new\src\main\java\org\cityu\group6\test\service\impl\ArticleServiceImpl.java
Generating \C:\JavaProjects\blogserver-new\src\main\java\org\cityu\group6\test\dao\ArticleDao.java
Generating \C:\JavaProjects\blogserver-new\src\main\java\org\cityu\group6\test\entity\ArticleDODO.java
Generating \C:\JavaProjects\blogserver-new\src\main\resources\mybatis\ArticleMapper.xml
java.lang.Exception: Can not find column information from table:tag
        at com.greedystar.generator.db.ConnectionUtil.getColumnInfos(ConnectionUtil.java:146)
        at com.greedystar.generator.db.ConnectionUtil.getMetaData(ConnectionUtil.java:69)
        at com.greedystar.generator.invoker.SingleInvoker.queryMetaData(SingleInvoker.java:19)
        at com.greedystar.generator.invoker.base.AbstractInvoker.execute(AbstractInvoker.java:86)
        at org.sang.utils.AutoCodeGenerate.main(AutoCodeGenerate.java:12)
```

Figure 23. code to invoke

Developers can select the tables in the database and input the corresponding format of the code, let the original tool automatically generate the corresponding data table class. This is useful, but it's easy enough. It's easy to get errors when developers need to generate many data classes. This also doesn't make it easy to review the code later. Moreover, it isn't automated enough either. Based on the GUI design, we made this function modification easier and more intuitive and more automatic.



Java Code Generator GUI

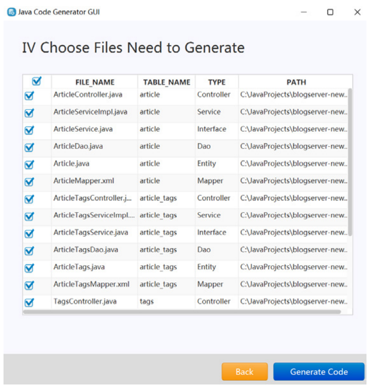
Ⅲ Choose Tables

☐	TABLE_NAME	CLASS_NAME	ROWS	CREATE_DATE
✓	article	Article	14	2022-11-03
✓	article_tags	ArticleTags	14	2022-11-03
☐	category	Category	6	2022-11-03
☐	comments	Comments	0	2022-11-03
☐	pv	Pv	8	2022-11-03
☐	roles	Roles	5	2022-11-03
☐	roles_user	RolesUser	17	2022-11-03
✓	tags	Tags	14	2022-11-03
☐	user	User	11	2022-11-03

Figure 24. choose tables

As we just said in step 3, the tool can show all the tables in the database. We store each table in the database.
Another selection function of the original tool is to select which files to generate. Continuing with the SpringMVC framework as an example, in the original tool, the developer needs to enter the following to generate the corresponding file:
path:
  controller: controller
  service: service.impl
  dao: dao
Of course, if developers don't want tools to generate controller-related files, they can just leave the content after the 'controller:' blank. But we don't think it's possible for the original tool to meet the requirement that the developer just wants to generate some of the files in the controller. We have also improved this part of the original tool to give the developer more flexibility to choose the files he wants to generate.

First, as with the table selection functionality, the tool will specify whether the developer wants the file to be generated by setting the selected parameter to True or False on the box object of the FileGenerationInfo class. And then, in the final code generation process, the tool will skip the generation of code and files that were not selected by the developer. These files which are not selected by the developer will not appear in the file validation step.
We think that selecting function changes give developers more flexibility in what code they want the tool to generate automatically. This makes the tool convenient and flexible.



Java Code Generator GUI

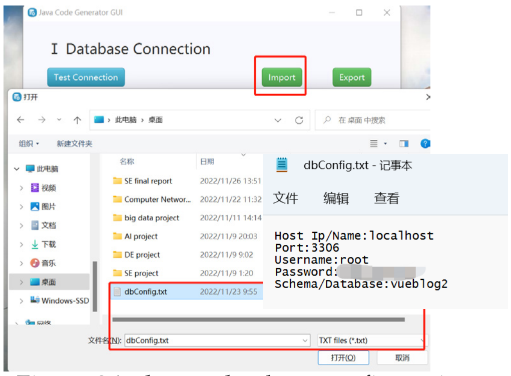
Ⅳ Choose Files Need to Generate

✓	FILE_NAME	TABLE_NAME	TYPE	PATH
✓	ArticleController.java	article	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleServiceImpl.java	article	Service	C:\JavaProjects\blogserver-new...
✓	ArticleService.java	article	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleDao.java	article	Dao	C:\JavaProjects\blogserver-new...
✓	Article.java	article	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleMapper.xml	article	Mapper	C:\JavaProjects\blogserver-new...
✓	ArticleTagsController.j...	article_tags	Controller	C:\JavaProjects\blogserver-new...
✓	ArticleTagsServiceImpl....	article_tags	Service	C:\JavaProjects\blogserver-new...
✓	ArticleTagsService.java	article_tags	Interface	C:\JavaProjects\blogserver-new...
✓	ArticleTagsDao.java	article_tags	Dao	C:\JavaProjects\blogserver-new...
✓	ArticleTags.java	article_tags	Entity	C:\JavaProjects\blogserver-new...
✓	ArticleTagsMapper.xml	article_tags	Mapper	C:\JavaProjects\blogserver-new...
✓	TagsController.java	tags	Controller	C:\JavaProjects\blogserver-new...

Figure 25. choose files need to generate

5.3.5 Save and read function

If we have two projects that need to use the same database, the original tool requires the developer to enter the database information twice for each project. We consider such an operation too repetitive and meaningless. So, to improve this problem, we added the ability to export and import database settings.
In the first step of the new tool, developers have the option to export the input database information and save it in a local file. The tool reads the database information that the developer has entered and pairs each paragraph with the database parameters it represents. And then, the tool will export the Database parameters in the format of a.txt file, the file content mainly includes the 'Host Ip/name', 'Port', 'Username', 'Password' and 'Schema/Database' these five attributes and the corresponding value of each attribute.



Java Code Generator GUI

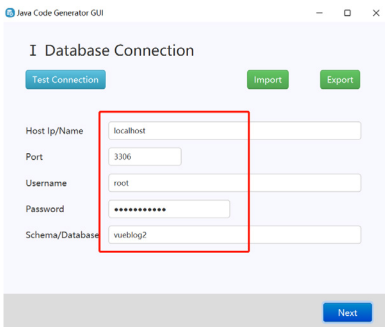
Ⅰ Database Connection

名称	日期
SE final report	2022/11/26 13:51
Computer Networ...	2022/11/22 11:32
big data project	2022/11/11 14:14
AI project	2022/11/9 20:03
DE project	2022/11/9 9:02
SE project	2022/11/9 1:20
dbConfig.txt	2022/11/23 9:55

文件名(N): dbConfig.txt TXT files (*.txt)

```
Host Ip/Name:localhost
Port:3306
Username:root
Password:[redacted]
Schema/Database:vueblog2
```

Figure 26. choose database configuration



Host Ip/Name: localhost
Port: 3306
Username: root
Password: •••••••••••
Schema/Database: vueblog2

Figure 27. successfully import database configuration

The read function is the opposite of the save function. In the first step, the developer can ask the generator to read the previously exported database settings file. We divide each line of the file into two parts based on the colon, the first half of which is the keyword, and the second half of which is the value corresponding to the keyword. Then replace the value in the corresponding text box according to the same keyword.

We think this addition will make the code generation tool much easier and more powerful.

According to the pain points we found in the process of using the original code generation tool, we improved some functions of the original tool and added some functions. We believe that these improvements are able to solve some of the problems in the original tool. This makes the tool simpler and more powerful.

5.4 Limitation

5.4.1 lack of error message

In some cases, the program will display a successful connection to the database, but tables have not been loaded by the program, but in this error case, there is no error message or program log, which is very difficult for troubleshooting. A certain difficulty.

5.4.2 Unable to customize CURD code

The program does not provide the function of generating conditional queries and aggregate functions. When such code is needed, it is still necessary to manually modify the code.

5.4.3 UI interface is not adaptive

In order to improve the usability of our tool, we optimized the UI interface to make it more user-friendly. However, there is still a small problem that our UI interface can be displayed completely on large-size computer screens, but it cannot be adapted on small-size computer screens, and some of the interfaces is not displayed completely. To solve the problem that some buttons cannot be clicked on small-size screens, we have made some improvements, such as adding a scroll bar. It basically solves the problem that the obscured buttons can't be clicked, but there is still the problem that they can't adapt to the screen size.

5.4.4 Can't support multiple programming languages

Due to the language limitation of the original project and the limitation of the programming language and development framework we use, our tool can only support the automatic generation of java code at the moment, and cannot support other languages such as Python, C++, etc. for the time being.

6. Conclusion & Creativity

This paper introduces the problem of repeated code writing and increased developer workload that may be encountered when using the Java framework for application development. Discusses the idea of using JAVA code generation tools to automatically generate code, improve development efficiency, and reduce the burden on developers, introduces the principles of existing JAVA code generation tools and their shortcomings, and how we build our project based on this. This paper also introduces how we use Scrum, an agile development tool, to scientifically manage projects, how to design software architecture, use JavaFX, Freemarker and other frameworks to develop our tools, and how to use QF-Test, an automated testing tool, to assist Coverage testing and evaluation of our tools.

In the specific field of Java code generation using templates based on Java frameworks such as Spring and Mybatis, we compared various existing tools and related papers, based on the idea of how to make tools simpler and more powerful, combined with the existing JAVA application development environment, the three optimization directions of implementing GUI, adding database-related functions, and adding file management-related functions were innovatively proposed, and finally completed our software engineering tool. And the practical value of the tool is verified by means of automated testing and packaging into executable files.

References

[1] Kopp, Andrii, and Dmytro Orlovskyi. "An Approach and a Software Tool for Automatic Source Code Generation driven by Business Rules."W.-K. Chen, *Linear Networks and Systems.* Belmont, Calif.: Wadsworth, pp. 123-135, 1993.

[2] Milosavljević, Branko, Milan Vidaković, and Zora Konjović. "Automatic code generation for database-oriented web applications." Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002.

[3] Meidinger, M., Ebbers, H., Reimann, C. (2015). AeroFX - Native Themes for JavaFX. In: Dregvaite, G., Damasevicius, R. (eds) Information and Software Technologies. ICIST 2015. Communications in Computer and Information Science, vol 538. Springer, Cham.

[4] G.Kruk, O.Alves, L.Molinari, E.Roux, CERN, Geneva, Switzerland. BEST PRACTICES FOR EFFICIENT DEVELOPMENT OF JAVAFX APPLICATIONS, 2017

[5] Quality First Software homepage, https://www.qfs.de/en/index.html

[6] Chen, Bing-Can. Research and Implementation of QF-Test-based Automated Functional Testing Framework. MS thesis. Shanghai Jiaotong University, 2012.

[7] David Harrison. Project-Based Test Automation. Testing Experience Magazine 2, June 2009