QuizScript 2: Improved Interactive Quizzes for the Masses

by

Yasser Haji Valizadeh
B.Sc., University of Victoria, 2008

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

QuizScript 2: Improved Interactive Quizzes for the Masses

by

Yasser Haji Valizadeh
B.Sc., University of Victoria, 2008

Supervisory Committee

Dr. William W. Wadge, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. William W. Wadge, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

ABSTRACT

In this thesis we present the improvements and refinements that we made to the QuizScript system. QuizScript allows users, using simple markup, to quickly and easily generate interactive quizzes, composed of multiple types of questions, and provides immediate feedback to students. It was designed by Christine and Bill Wadge to overcome the tediousness often experienced while creating quizzes using other systems, such as Moodle. However, QuizScript has several shortcomings and their rectification is the focus of this thesis. We remedied these drawbacks by designing and implementing the following: a parser to detect errors; new functionalities; a database to store and retrieve the quizzes; and a dynamic website to host the quizzes, which enables an immediate preview of the quiz.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:
My parents, Ozra and Mohammad, for their support.
My supervisor, Dr. Wadge, for his guidance.
My editors, Rakhi Bhatnagar, and Romina Falbi Franzoni, for their insights.

DEDICATION

This is dedicated to my brothers, Sina and Behzad. Hope to read your theses some day soon.

Chapter 1

Introduction

Quizzes and exercises are an integral aspect of education. It is not surprising that instructors are increasingly using Learning Management Systems (LMS) to deliver course material, including quizzes, and making them accessible to the public [1]. There are numerous different LMS that provide a complete computerized or on-line teaching experience, such as Moodle. In addition to delivering learning materials, managing users, and providing communication channels, they also provide tools for assessment. These assessment tools, however, take a very long time to create a simple quiz, and require many interactions with the instructors. Even though Moodle was ranked the best open source LMS in 2005 [2], making quizzes on this system is tedious. An irritating amount of repetitive tasks is required of the instructor to generate a simple quiz, composed of only a few multiple-choice questions. Also, while answering the quizzes, students do not receive immediate feedback until after submitting the entire quiz. These drawbacks were noticed by the Wadges and inspired them to create a simple-to-use system: QuizScript.

QuizScript is a system that makes the process of generating interactive quizzes simple. Specifically, quizzes composed of multiple-choice and fill-in-the-blank question types. There is no software to download or install, both on the instructors' and the students' machines [3]. The only requirement to create a quiz is to have an internet connection and web browser that supports JavaScript.

The content of the quiz can be typed or simply copied and pasted from another source, such as a newspaper article. In order for QuizScript's engine to create the questions, the instructor has only to punctuate the text. Punctuating the text is as

simple as inserting a few 'Trigger Symbols'[1] in the right places. For example, inserting the equal sign, "=", before any word, will result in a fill-in-the-blank question with the word being the answer.

The generated quiz is simply an HTML page. This page is embedded with JavaScript, which provides the interactivity and checks for the correctness of the answers. The quiz can be saved using the browser's "Save as" functionality and, later, uploaded to a server or distributed to students by any method of transference. The only requirement for the students is opening the file with a browser which supports JavaScript. Other than that, QuizScript is simple to use and accessible from any device that has a JavaScript-enabled browser.

QuizScript boasts that anybody, after following a short tutorial, can easily and quickly create unique and innovative quizzes within a few minutes.[2] It also provides immediate feedback to students after each question. With that said, QuizScript has a few shortcomings and improving these flaws became the motivation of our thesis. This chapter presents the major problems of QuizScript, the solutions implemented, and concludes with an outline of the remainder of this thesis.

1.1 Problems and Motivations

Despite the ease-of-use of QuizScript, there exists a number of glaring obstacles. First, it can be assumed that instructors will make honest mistakes while generating quizzes on QuizScript. The system, unfortunately, responds with a vague explanation that leaves the user confused. Second, inserting HTML into the script confuses the system, and subsequently the user. Third, there is no way of telling the system not to parse the Trigger Symbols. Finally, QuizScript does not store or host the quizzes for retrieval and distribution purposes.

In an attempt to simplify the quiz-making process, QuizScript requires users to insert special characters, Trigger Symbols, at the appropriate location. QuizScript's engine automatically detects these Trigger Symbols and generates the appropriate question. The problem arises when the user does not insert the Trigger Symbol in the correct place or forgets to follow the rules governing that particular Symbol. This

[1] Trigger Symbols are: =, #, (, |,), @, .c, and *.

[2] On February 2010, I accompanied Bill Wadge to Tapestry Conference held at Esquimalt High School, in Victoria. Here we presented QuizScript to a classroom of teachers, explaining how to create quizzes and the required punctuation. Immediately after our presentation, we observed the teachers were quickly and accurately generating quizzes on their own using QuizScript.

prompts the system to display an extremely vague error message. It does not provide any useful information to the user to either locate or correct the error, and only people familiar with the design of QuizScript, or with a computer science background, would be able to decipher it. To rectify this problem, I created a parser that scans the user's input and locates potential mistakes. If there are mistakes present, the parser will indicate both the location of the errors and provide helpful suggestions on how to correct them.

An appealing feature of QuizScript is that it allows instructors to insert HTML tags inside the script. Users familiar with HTML can create links within the quiz, insert pictures or videos, and create tables or use other capabilities of HTML. Unfortunately, this can confuse QuizScript because it will try to parse the parts of the HTML tag, which use Trigger Symbols, into a question. To resolve this problem, I implemented a feature to the parser which detects HTML tags and instructs the parser to skip over them. QuizScript will then not modify the contents inside the tags.

Similarly, QuizScript struggles when instructed to include text that uses a Trigger Symbol. This could occur in a question that involves the following phrase: "1+2 =3." Currently, as mentioned, QuizScript does not have a mechanism to ignore and skip Trigger Symbols. In such a case, QuizScript will automatically turn the text into a question. To fix this drawback, I added another feature to the parser that instructs it to skip over any desired character. To do this, the user simply has to insert a backslash, "\", before the chosen character.

Finally, QuizScript does not host or store the quizzes, forcing instructors to manually distribute them. Upon successfully creating a quiz using QuizScript, the teacher can then copy the file and distribute it using any medium: CD, thumb drive, e-mail, a post on their personal website, or any other method of transferring files [3]. These methods, however, are not a feasible or effective means to distribute the quiz to students. Transferring the file to each student's CD or thumb drive is time consuming; also, not every instructor has their own website. To overcome this distribution bottleneck, in conjunction with a database and web server, I created a dynamic website to host the quizzes. A unique URL is created for each quiz which can then be accessed from any browser. Currently, QuizScript also does not store the quizzes for later retrieval. Unless the instructor explicitly saves the created quiz on their computer's hard drive, the quiz disappears as soon as the browser is closed. To address this problem, I created a database to store the quizzes. The stored quiz can then be

retrieved, edited, or viewed at a later time.

1.2 Thesis Outline

Following this section will be a brief history of LMS. Then, in Chapter 3, there will be an in-depth description of the Wadges' QuizScript. Chapter 4 will explain the core of my work continuing to Chapter 5 outlining the implementation of the parser. The thesis will, then, conclude with a summary of my work.

Chapter 2

Quiz Generating Software

In this chapter we look at the development of Learning Management Systems (LMS), beginning with the earliest form in 1924. Following this, we explore current examples of LMS available. Finally we focus on one LMS example, Moodle, to show the tediousness of making a quiz using its quiz-generating process, which served as the motivation for the creation of the Wadges' QuizScript.

2.1 Development of Learning Management Systems (LMS)

In this section we begin with an abridged summary of the evolution of LMS. Essentially, the purpose of LMS is to provide study materials for students, enable a conversation between the instructor and student, to administer exams, and report scores.

The first teaching machine was invented in 1924 by Sidney L. Pressey, an educational psychology professor at Ohio State University [4, 5]. Pressey created this machine to provide drill and practice items for his students [6]. The form of this early device was modelled like a typewriter, and was capable of administering multiple-choice tests. Through a window in the machine a question would appear along with four potential answers for the student. Such devices moved through questions systematically only after a correct answer was chosen. For Pressey, it was this aspect of the device that made it a "pedagogical tool," as the student had to "learn" the proper answer before turning to the next question [4]. According to Skinner, the Edgar Pierce Professor of Psychology at Harvard University until 1974, "Pressey seems to

have been the first to emphasize the importance of immediate feedback in education and to propose a system in which each student could move at his own pace [4]." This revolutionary aspect, unfortunately, has not always been incorporated in more recent LMS, like Moodle.

Later in the century, around the 1980s, personal computers began to appear in households around the world. As computers became more and more popular, various teaching and testing software and hardware were created. Before the internet, to access e-learning materials, students had to physically travel to an institution and sit in a kiosk [7]. With the advent of the internet, and the continued growth of personal computers, the number of teaching computer programs increased. Along with this, the user no longer had to physically be present at the kiosk.

With the help of the World Wide Web (Web), the method of distributing learning content shifted notably – from a specified physical kiosk to a personal computer located anywhere. This physical shift, and the accessibility of the Web, resulted in the increased popularity of on-line learning, driving the need for improving and creating more LMS [7]. In "A Critical Examination of the Effects of Learning Management Systems on University Teaching and Learning" by Coates et al. [8], the authors discuss the development of LMS since the 1990s. According to them, the improved systems have been embraced by universities globally, and are referred to by the following terms: course management systems, learning platforms, content management systems, distributed learning systems, portals, and instructional management systems. These systems are capable to "drive virtual universities [8]." At the same time, all LMS differed greatly from each other. Some systems boasted numerous assessment possibilities, while others offered only a few options [8, 7], and it was impossible to share content between any of the systems. These drawbacks drove the need to develop a set standard for LMS.

A standard had to be made, and in 1988 Aviation Industry Computer-Based Training Committee put forth a standard that became the de facto [7]. As the standard was mostly developed prior to using the Web to deliver content, another version needed to be made. The Department of Defense, in the USA, led the way to create a standard that could be used by any LMS. In October of 2001, a robust and stable version of SCORM (Sharable Content Object Repository Model) was released and it was quickly adopted by the industry, for it allowed for the easy transfer of content between systems. Even though it has gone through a few revisions since its first release, a lot of LMS support a version of it [7].

Standardization allowed for instructors to transfer on-line courses between different LMS. In the past decade, as mentioned, universities have been quick in adopting and using a form of LMS. In fact, statistics show that there has been an explosion in the use of LMS in universities worldwide. Between 2000 and 2008, the percentage of higher-education courses that use an LMS increased steeply from 14.7% to 53.5% [9]. By 2005, 70% of Australian, UK, and Canadian universities used an LMS [8]. Only four years later, it is reported that 97.5% of universities in the US used an LMS [10]. This trend is expected to continue to grow as higher-education establishments are starting to require and encourage a form of web presence in their courses. As expected, this trend has moved into secondary education as well, an example of which is found in Ontario with around 100 courses offered on-line [11].

Essentially, the main responsibilities of most LMS can be categorized as:

- *User management:* create users and assigning roles and responsibilities to each.
- *Content delivery:* ability to deliver the learning material to the user, most commonly through the Web.
- *Communication:* provide different channels for communication, for example most LMS support chat, blog, email, and forums.
- *Assessment:* the ability to deliver assessments and scoring the learner.

Based on the previous statistics, it is not surprising that the LMS industry has grown close to a billion dollar industry a year. As of May 2010, there is a total of 301 SCORM-certified products [12] and we will list a few of them in the following section.

2.2 Current Examples of LMS

As mentioned, there are around 301 SCORM-certified products available.

List of LMS:

- Blackboard
- Moodle
- Desire2Learn
- Atutor

- Udutu
- Sakai

In 2011, Blackboard was the leader in the higher-education market with 51% market share. After this came Moodle and Desire2Learn with 19% and 11% market share respectively [13]. With over 70 million registered users and about 200 million quiz questions [14], Moodle is the leading open source LMS.

2.3 Focusing on Moodle

While working as a web-master and a system administrator at Curtin University of Technology in 1990's, Martin Dougiamas became frustrated with the university's LMS, WebCT, which has now been bought by Blackboard [15, 16]. He realized that all the current LMS were designed by engineers and not by educators. He wanted a system that was easy to use, and designed by instructors for instructors. Dougiamas wanted a system that would be "free" and allowed teachers to move their teaching skills on-line:

> "My strong beliefs in the unrealised possibilities of Internet-based education led me to complete a Masters and then a PhD in Education, combining my former career in Computer Science with newly constructed knowledge about the nature of learning and collaboration. In particular, I am particularly influenced by the epistemology of social constructionism - which not only treats learning as a social activity, but focusses attention on the learning that occurs while actively constructing artifacts (such as texts) for others to see or use."
>
> Martin Dougiamas [15]

As a result of his research and work, Moodle (Modular Object Oriented Development Learning Environment) was born. The first stable version of Moodle was released on August 20th 2002. The entire project is released under an open source license, which allows anybody the freedom to modify, change, distribute, and obtain the source code.

Moodle is being used by universities, high schools, government organizations, private companies and even home-schooling parents [15]. Ever since it was first released, with a user base of only 1 instructor, Peter Tyler (Dougiamas' supervisor), and his

students [17], it has grown to a complete platform with an astonishing number of people that use it. Currently, there are more than 1.2 million teachers and over 70 million users. In addition, the system currently has about 200 million quiz questions [14].

Moodle is a full feature LMS that includes tools for creating forms of assessment. Instructors are able to create a range of questions: multiple-choice, True/False, Numerical, Matching, Description, etc. With quizzes and exercises being essential in educational systems, we now explore how to create simple quiz on Moodle.

2.3.1 Process of Quiz-Making in Moodle

For example, suppose an English instructor has just taught a class on Possessive Adjectives (my, your, his, her, its, our, their, whose) and would like to create a quiz composed of a few multiple-choice questions to test or reinforce the students' knowledge on the subject matter.

The desired two multiple-choice questions, and their answers, are:

1. I am writing my book.

2. You are writing your book.

The teacher intends for the underlined words to be replaced with a "blank", allowing the students to choose from the following choices: my, your, his, her, its, our, their, and whose.

There are 2 options to make this quiz in Moodle:

Option 1: create each question individually then later merge them together in one quiz. The following steps and diagrams illustrates how to make questions individually. In the example some tasks are omitted, such as logging-in, creating a link for the quiz in a specific course, and setting other parameters.

Step 1: Set up the quiz for the specific course, and start adding questions by clicking "Add a question":

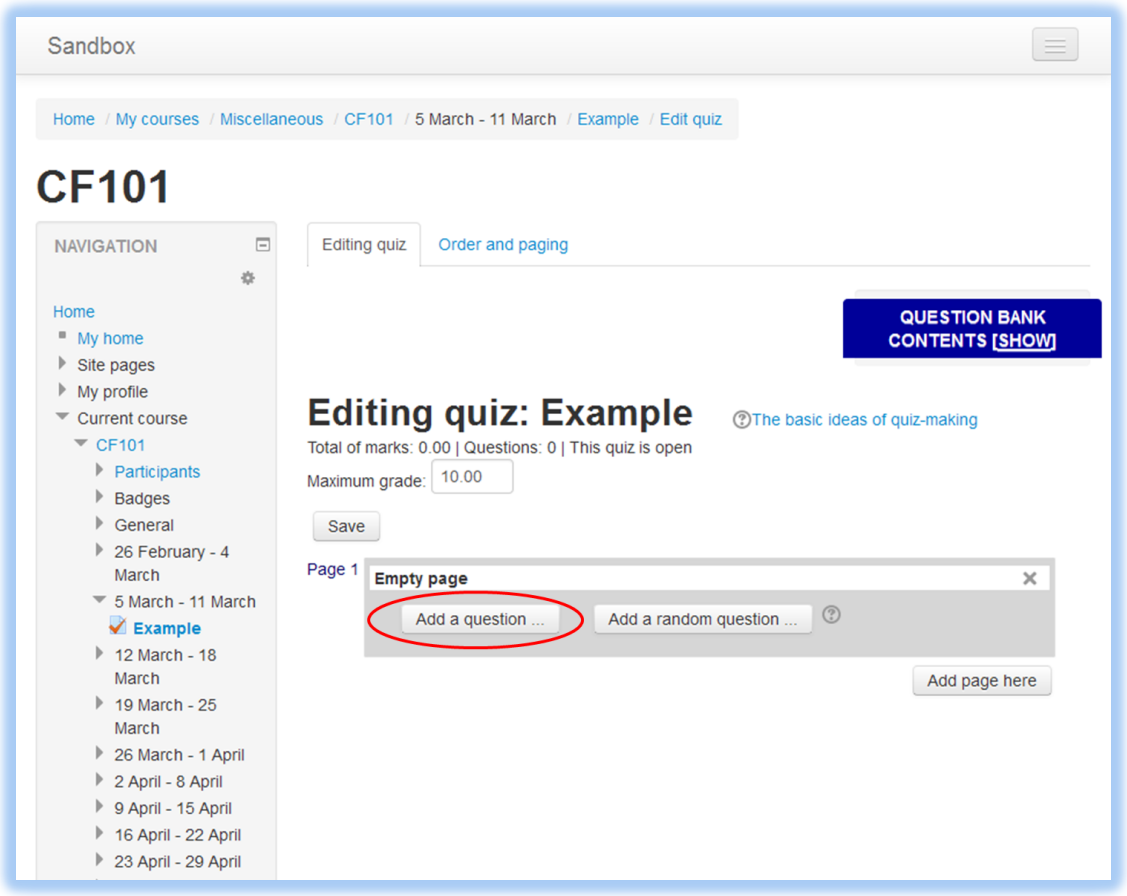

Figure 2.1: Adding a question to the quiz

Step 2: Select multiple-choice among the choice of question types and click “next”:

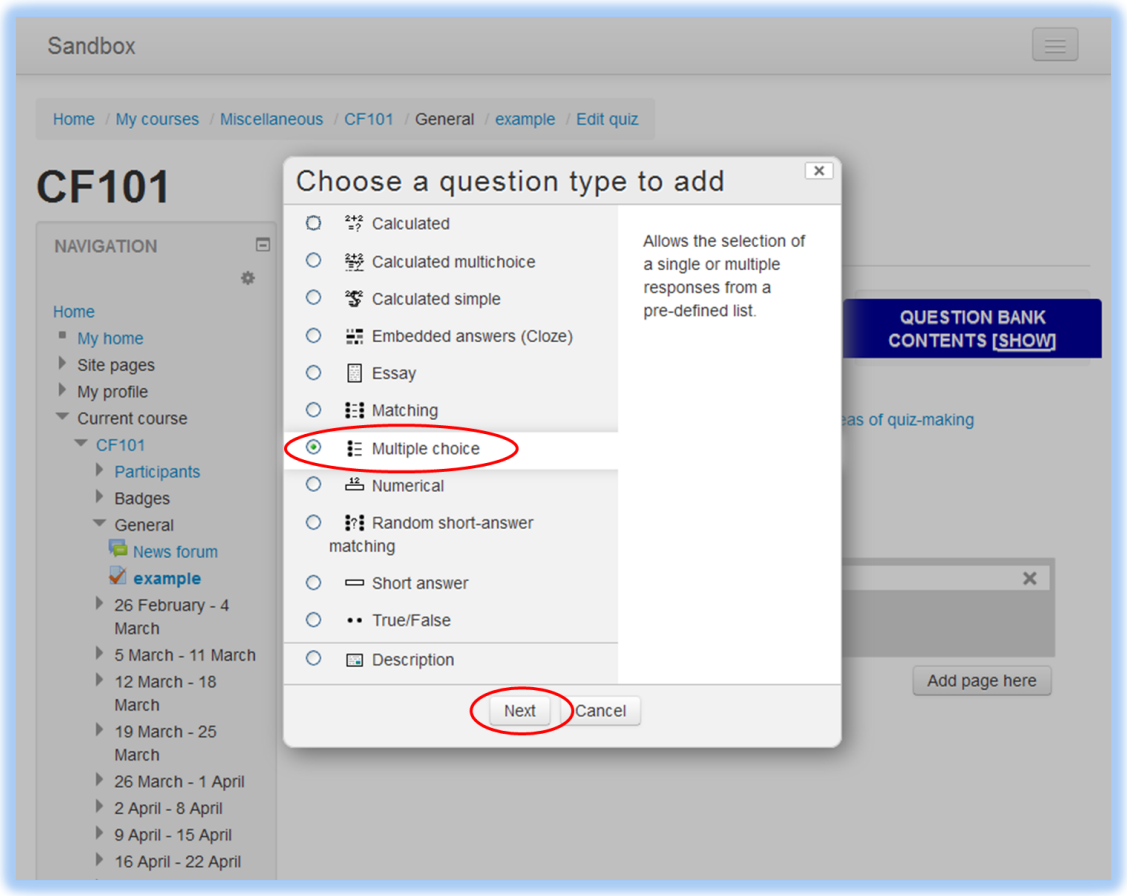

Figure 2.2: Selecting multiple-choice question among the list of question types

Step 3: Provide a name to the question and type the first question in the "question text" area:

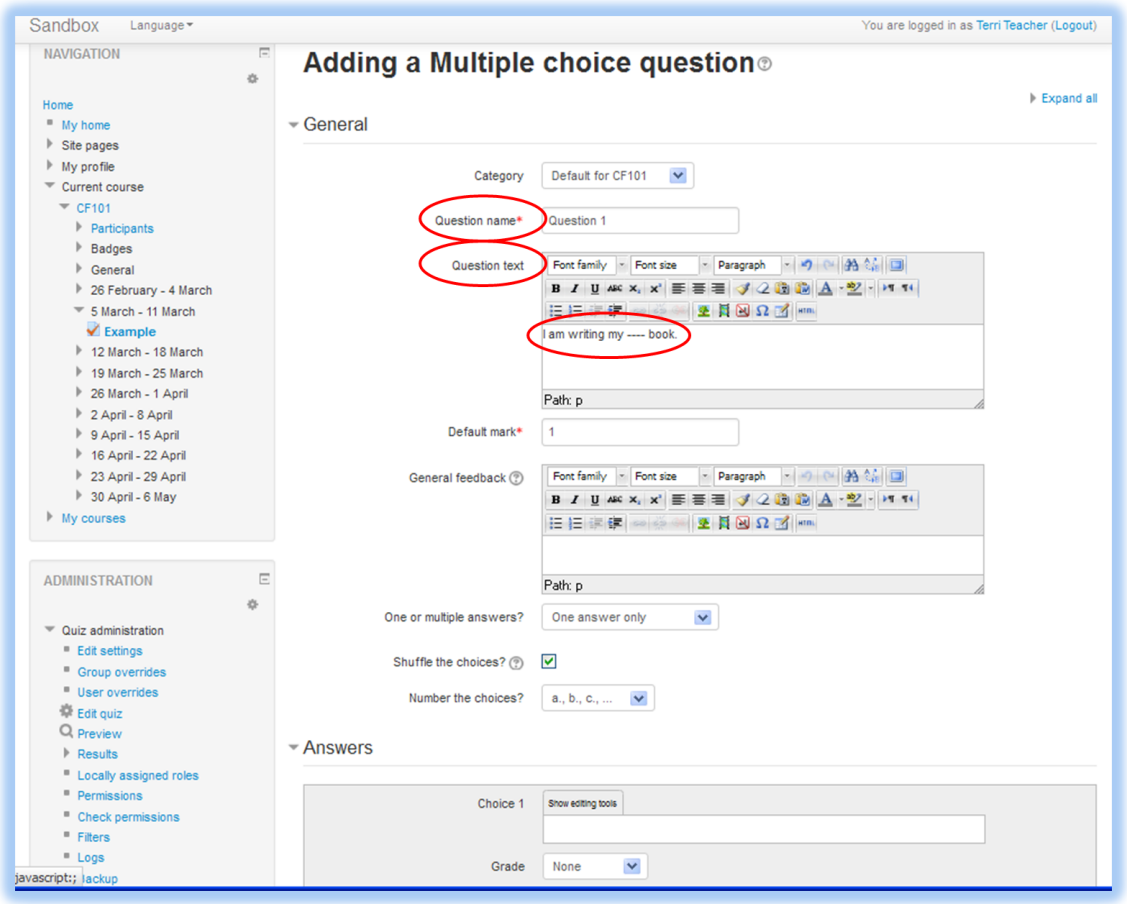

Figure 2.3: Creating the first question

It can be noted that one cannot insert a blank field where the choices are chosen from. In this example,"___" is used to emulate a blank area.

Step 4: Type each of the Possessive Adjectives in the areas provided for the choices:

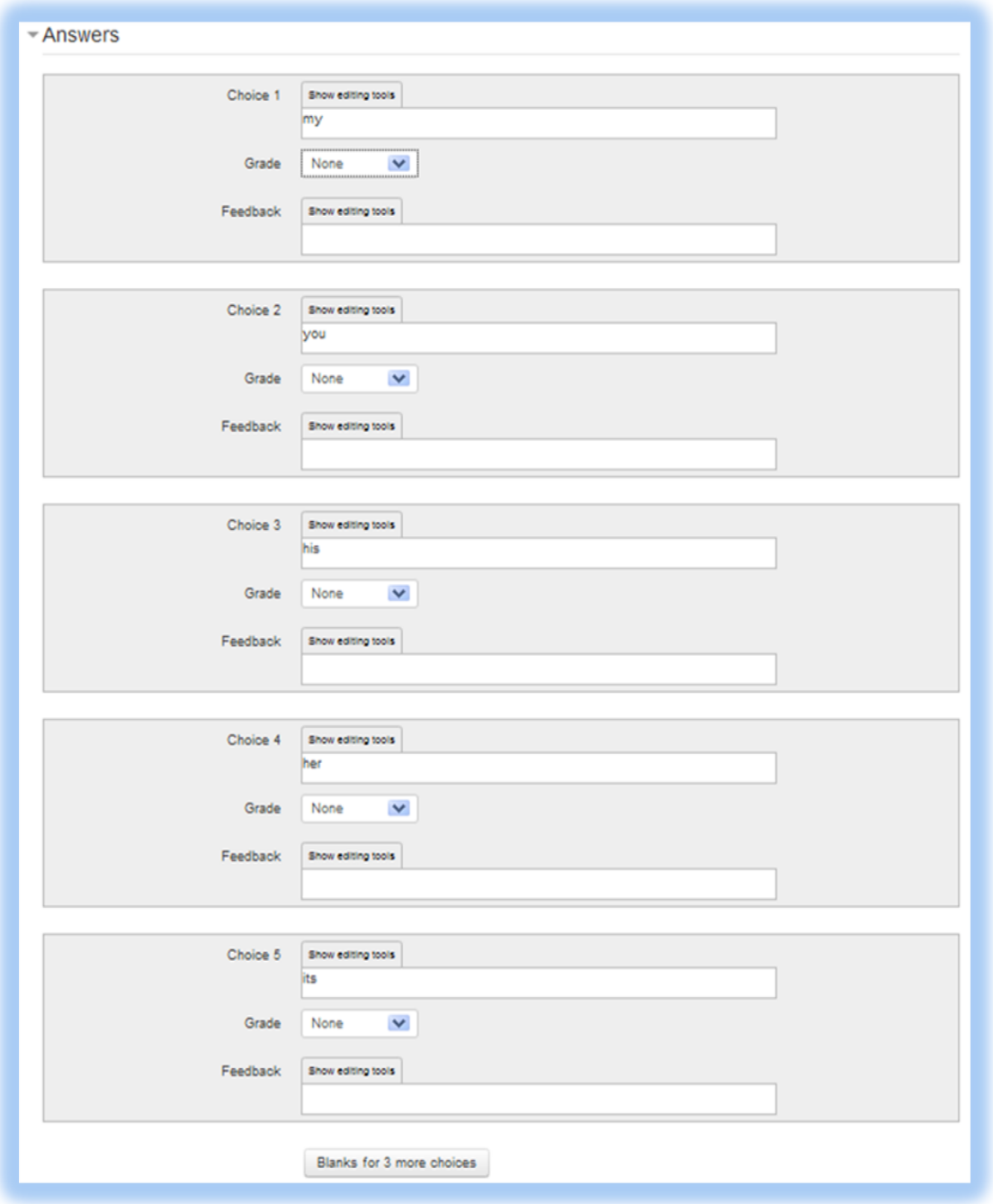

Figure 2.4: Inserting choices for the question

Step 5: At this moment we have exhausted all the available spaces for the choices and need to create more by clicking the "Blanks for 3 more choices" button at the bottom of the page.

Step 6: Repeat step 4 for the rest of the choices: our, their, and whose.

Step 7: Select the correct answer among the given choices. In this case we select the choice containing the word "my" and set its grade to 100%:

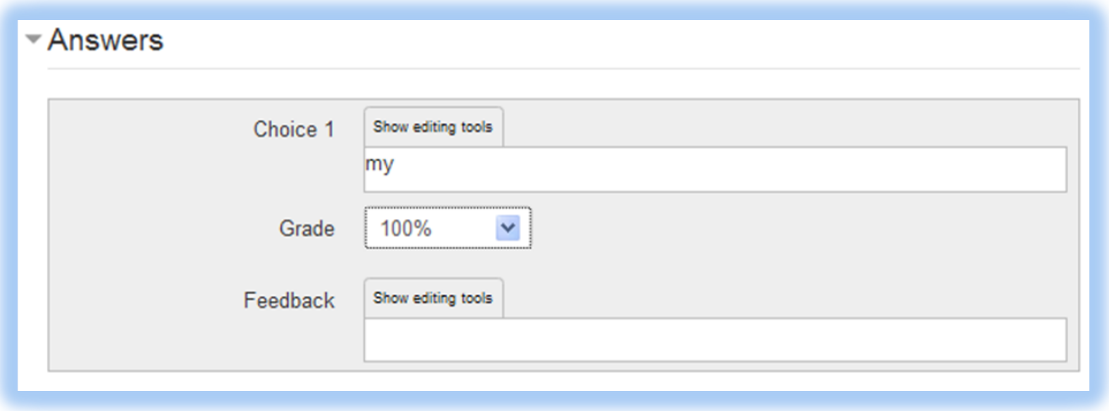

Figure 2.5: Selecting the correct choice among the list of choices

Step 8: Upon clicking the "Save" button, the user is taken back to the quiz page where the generated question can be previewed. A small magnifying glass underneath the question link needs to be clicked to open the preview window.

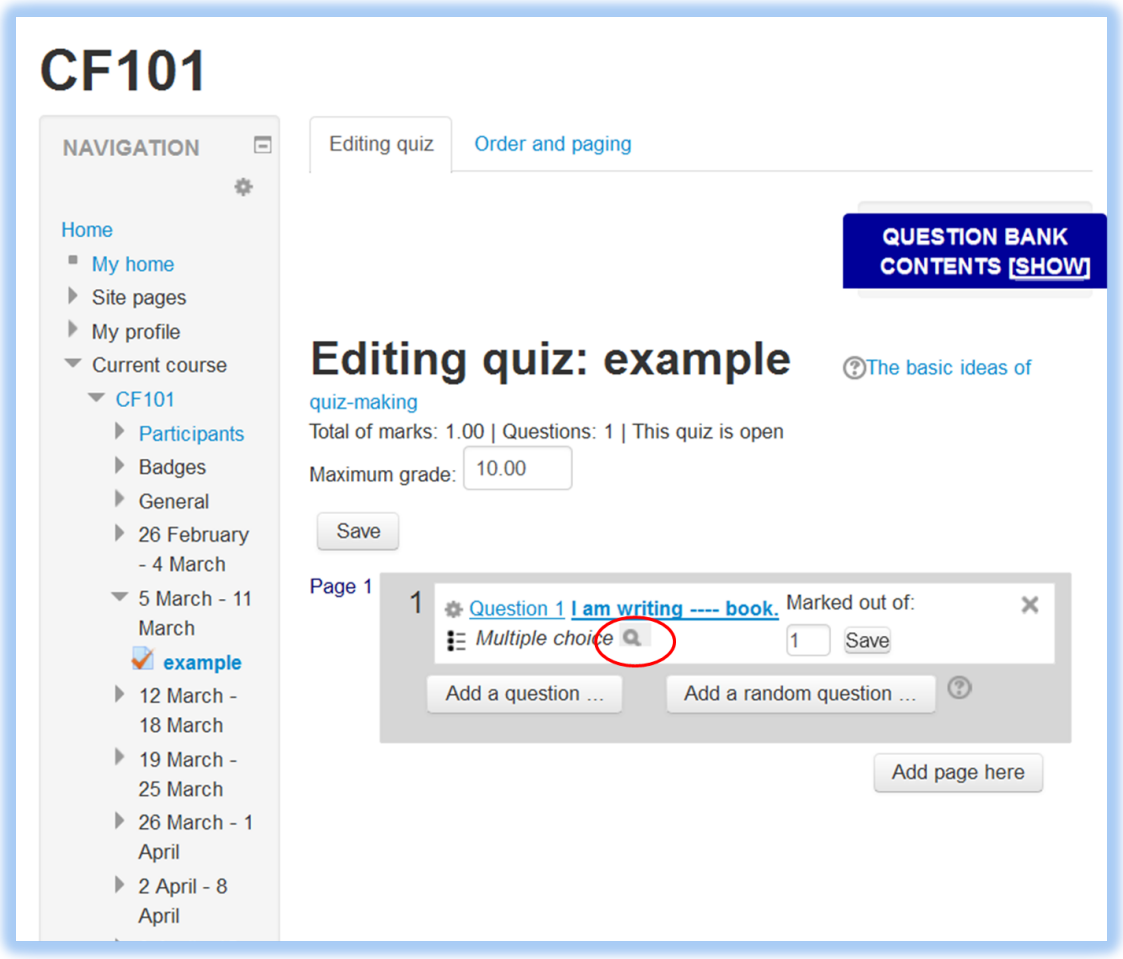

Figure 2.6: One question added to the quiz and available for reviewing

Step 9: The preview window shows how the student would see the question:

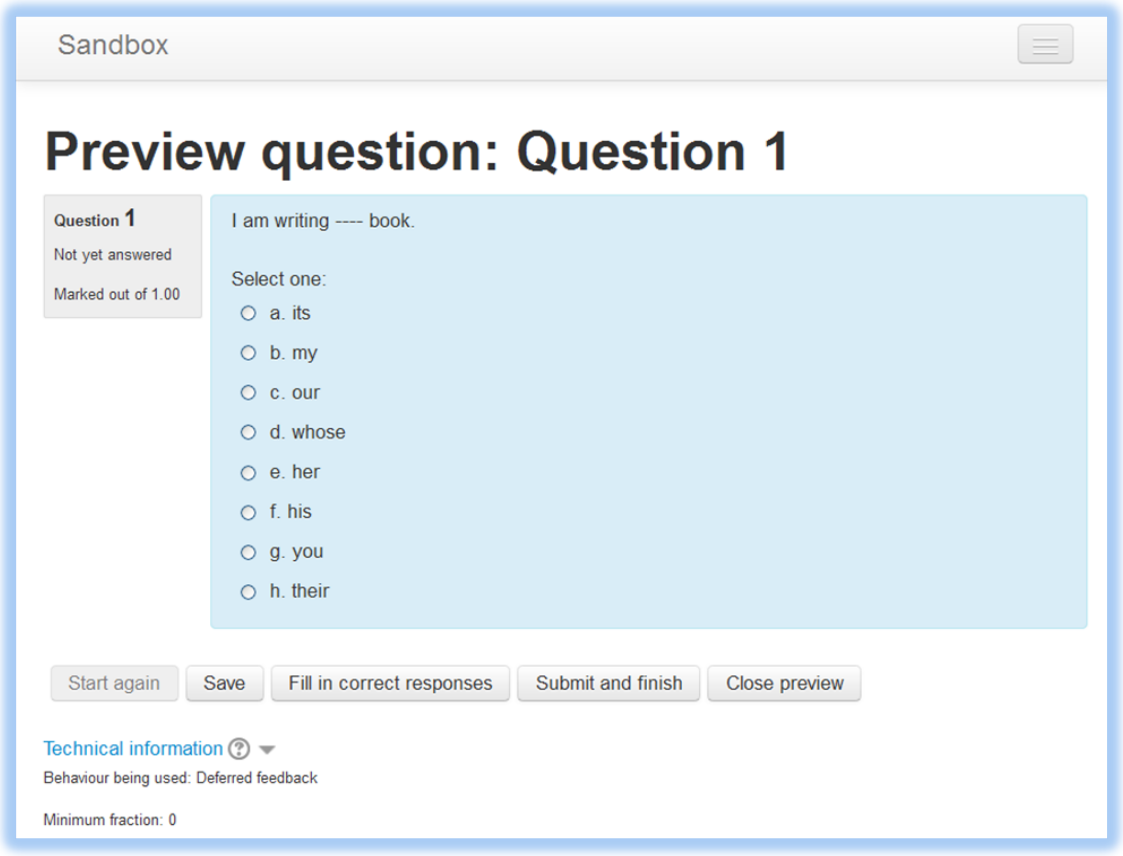

Figure 2.7: Preview of the question

It is worth to mention that the system does not provide any immediate feedback as the student is choosing the answer. Rather, the student needs to answer all of the questions, click the "Submit and finish" button and wait for the server to respond. The following diagrams show the system's responses to the correct and incorrect choices:

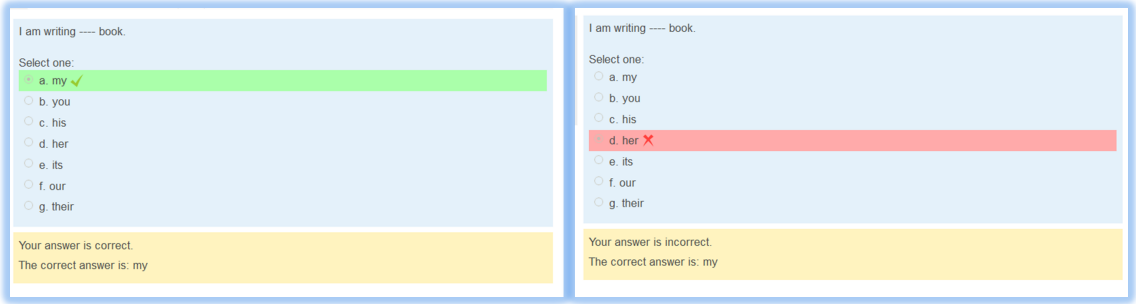

Figure 2.8: The result of choosing correct and incorrect answer

Even though we have simplified the process and omitted modifying numerous options and parameters along the way, there are 10 steps to follow in order to just make one question. The documentation provided by Moodle for this type of question requires 15 steps [18]. In order to create the second question the instructor is required to follow all the above steps again.

Option 2: use the "Embedded answers" or "Cloze" question type.
Unlike other types of questions provided by Moodle, the Embedded-question type does not have a graphical-user-interface. The user is required to insert special characters and symbols to create the questions [19]. The strict format and rules governing the set up of these types of questions could be overwhelming for users. Admittedly, in a document on its website, Moodle recommends users to use an alternative software to create such questions: Hot Potatoes. Moodle writes that "Hot Potatoes software is the easiest way to create Embedded answer (Cloze) questions"[19]. However, Hot Potatoes is a third-party computer program that must be downloaded and installed on the user's computer. The user can then create the questions and later import them back into Moodle.

Embedded-type questions allow for creating multiple types of questions (multiple-choice, short answers, and numerical answers) and the questions can be placed anywhere in the body of the quiz. In order to create the 2 questions in our example, as shown in the following diagram, a user would type:

- I am writing {:MC:=my your him her its our your their whose} book.
- You are writing {:MC:my =your him her its our your their whose} book.

In the question text area of Cloze question:

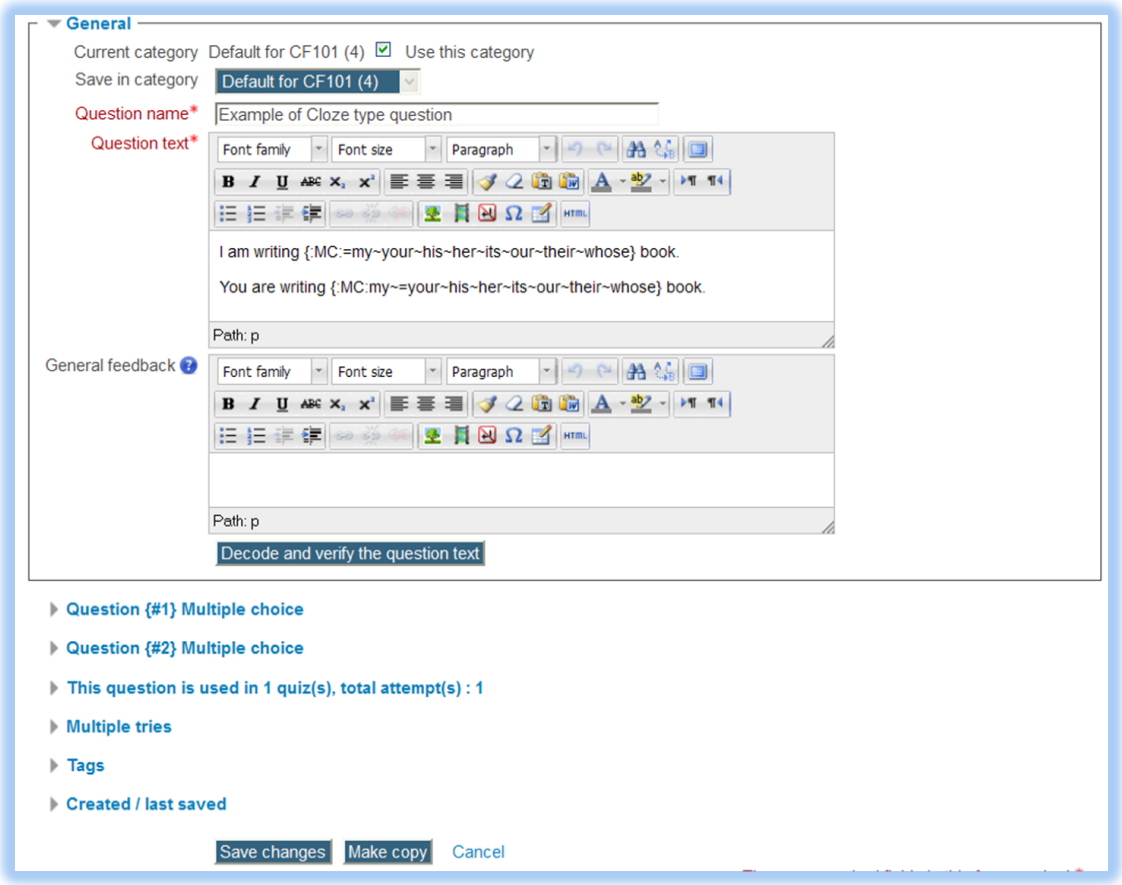

Figure 2.9: Creating the questions using "Cloze" type question

If the typed text is in the correct format, the desired question is generated, as shown in the following diagram:

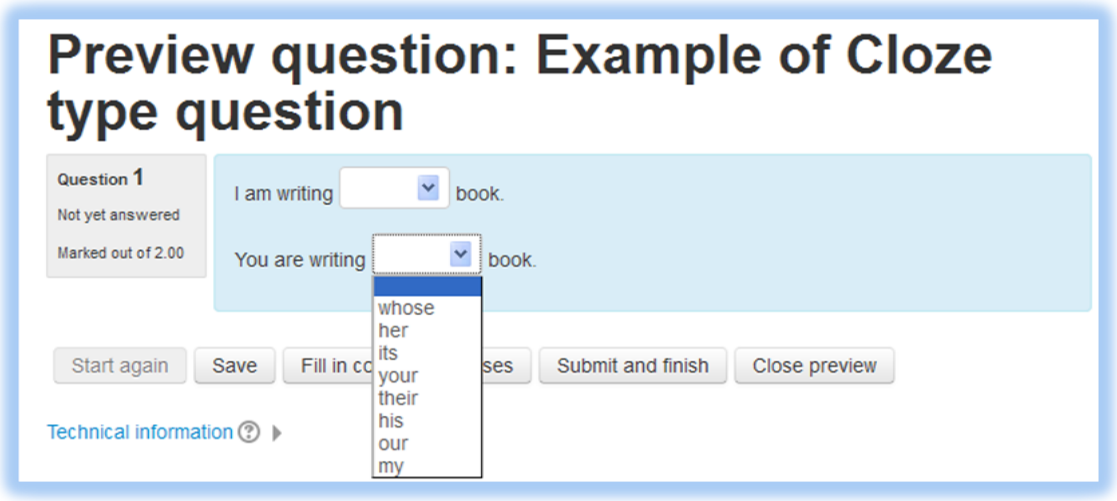

Figure 2.10: Questions created using "Cloze" method

This exploration shows the tediousness of making a quiz using the quiz-generating process on Moodle. As seen, to create only two simple questions, it requires at least 20 interactions with the teacher. Unfortunately, Moodle does not provide immediate feedback to the students even though the effectiveness of this feature was noticed by Pressey as early as 1924. To simplify this process, and provide immediate feedback, the Wadges designed QuizScript.

Chapter 3

QuizScript

The time consuming process of making quizzes on LMS and their lack of immediate feedback to students was what inspired the Wadges to create QuizScript. It is designed to be quick and easy to use. Creating a multi-questioned interactive quiz can be as simple as pasting a newspaper article and inserting just a few characters. This process can be mastered within a few minutes after reading a short tutorial. Also, on QuizScript, students benefit from immediate feedback to each question, and do not have to wait until the quiz is completed to identify their errors. Furthermore, unlike Hot Potatoes, another quiz-making program, QuizScript does not have any software to install as the entire process of generating and using the quiz occurs on the Web.

In this chapter we provide a detailed description of QuizScript. It begins with a thorough explanation of how to generate quizzes on QuizScript, and highlights its hidden features. After this, we compare QuizScript to Moodle, using the question sample from the previous chapter, to show that it is easier to make a quiz in QuizScript than in Moodle. The chapter then concludes by outlining the shortcomings of QuizScript.

3.1 The Details of QuizScript

Upon visiting QuizScript's website, the instructor is presented with a text box at the top of the page and a tutorial, a summary of the notations, at the bottom [20, 3, 21]. The tutorial guides the user through the steps of creating the quiz. It lists each possible question type and functionality, including "live examples" for clarification, followed by the notations required to generate that specific task. To create the quiz, the user then inputs the desired quiz contents into the text box above the tutorial,

including the appropriate notations. Once the input is submitted, after clicking the "Submit" button, QuizScript will recognize the notations and generate an HTML file that includes all the required materials to run the quiz. The deceivingly small text box can, in fact, accept very large files and inputs allowing for lengthy quizzes [3].

Quizscript Entry Page

Enter the quizscript source into the text area and press submit. This page will be replaced by a page presenting the specified quiz. You can take the quiz or save the page (as html source) for later use.

Submit

Read a short summary of the Quizscript system; try out a collection of exercises generated using Quizscript; consult the table below for a summary of the notation. If you have any questions contact us at cwadge@uvic.ca (Christine Wadge, French Department) or wwadge@uvic.ca (Bill Wadge, Computer Science Department) here at the University of Victoria.

Quizscript notation summary

What you want	What you write	Result
Fill-in-the-blank questions	Il a perdu =ses clés.	Il a perdu [1] clés. (Typing "?" into the blank will reveal the answer)
A clue in form of a word which lights up on mouse over	*Pierre* a perdu ses clés.	Pierre a perdu ses clés.
A pop up hint associated with a word or phrase	le repas est (délicieux|delicious).	le repas est délicieux.
Multiple choice questions from a particular list	Il a perdu [son_clé @ses_clés].	Il a perdu __[2].
Specify a list of choices for multiple questions to be used through out the quiz	.c mon ma mes .c1 son sa ses	
A multiple choice question from the global list.	J'ai perdu #mes clés. Il a perdu 1#ses clés.	J'ai perdu __[3] clés. Il a perdu __[4] clés.

Figure 3.1: QuizScript's homepage

In the provided text box, the instructor can type the content of the quiz or copy and paste it from another source, such as an on-line newspaper article. Once the content is inserted or typed it needs to be punctuated. The punctuation process is extremely simple [20] and it only requires inserting a few Trigger Symbols (Trigger Symbols are: =, #, (, |,), @, .c, and *) in the appropriate places. Creating different

kinds of questions and the description and usage of these Trigger Symbols are outlined and explained as follows:

Fill-in-the-Blank Questions:

The Trigger Symbol "=", an equal sign, is used to create fill-in-the-blank questions. Inserting an equal sign before any word, omitting the space, will create a fill-in-the-blank question where the chosen word is the answer.

Usage:

For example, "The capital of Canada is =Ottawa," will result in a question where the word Ottawa is replaced with a blank space in which students have to type the answer:

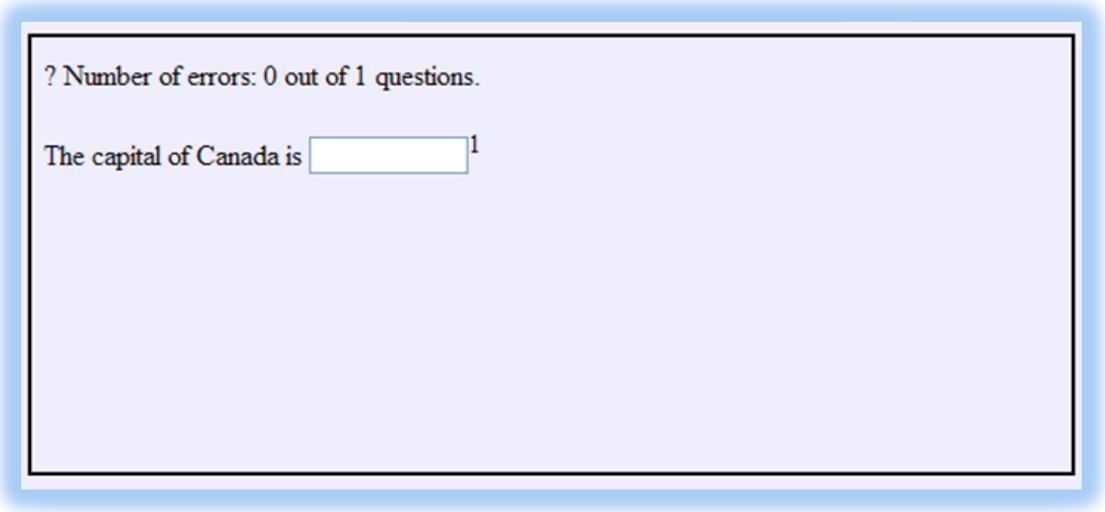

Figure 3.2: Example of Fill-in-the-Blank

Feedback: If the correct answer "Ottawa" is typed, then the word will replace the box and remain on the screen:

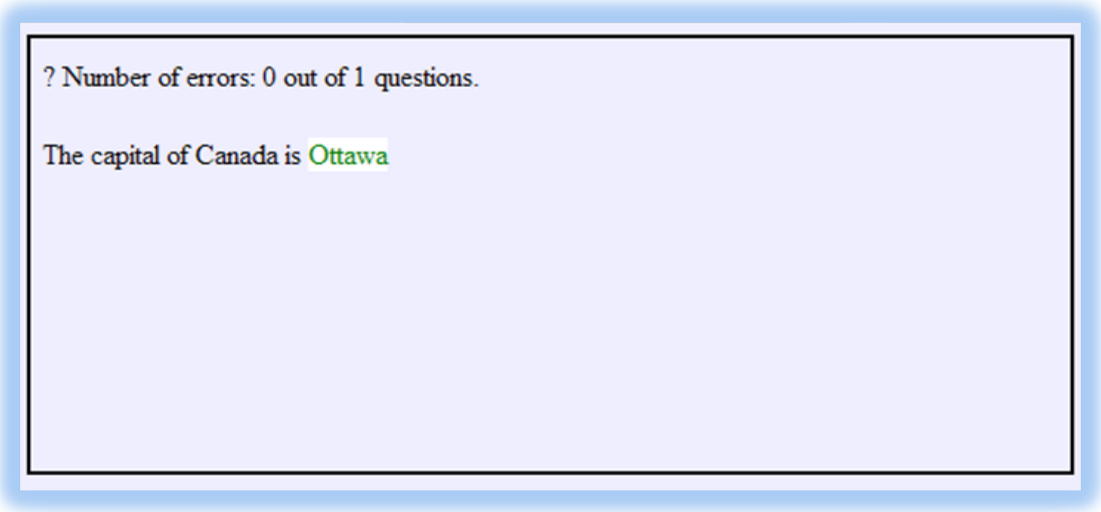

Figure 3.3: Fill-in-the-Blank question answered correctly

However, if an incorrect answer is typed, and the user presses enter, then the box deletes that input and remains empty and will add to the number of wrong tries which is shown on top of the page. The following diagrams shows the same questions attempted incorrectly multiple times:

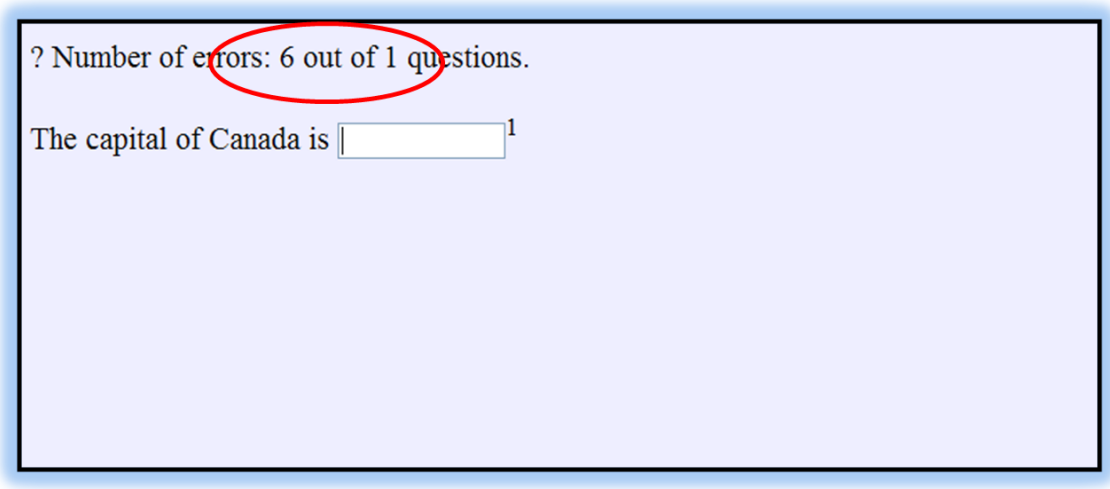

Figure 3.4: Fill-in-the-Blank question attempted incorrectly 6 times

In the chance that a student wishes for help, they can type a question mark (?) into the box to reveal the correct answer.

Multiple-Choice Questions:

QuizScript allows for multiple-choice question types to be inserted anywhere in the body of the quiz. There are two ways one can create such questions: "global" or "in-line":

Global multiple-choice Questions:

The instructor can specify a list of choices that can be used repeatedly throughout the quiz. Declaring the choices is as simple as starting a line with the Trigger Symbol ".c" followed by the choice options. One can even add a digit to the Trigger Symbol and create as many as ten distinct lists of choices.

Usage:

For example, to use two different lists, the days of the week and the different seasons, as the possible list of choices that will appear in the quiz, one has to type:

.c Monday Tuesday Wednesday Thursday Friday Saturday Sunday

.c1 spring summer fall winter

To use or create a multiple-choice question, all the instructor needs to do is type the question and insert the Trigger Symbol "#" before the answer. Also, if a digit was added to the Trigger Symbol ".c", then the same digit must precede the "#" symbol. Each list of answers can be repeated throughout the quiz, which in turn saves time for the instructor creating the quiz. For example: the first working day is #Monday. The hottest season of the year is 1#summer, and 1#winter is the coldest.

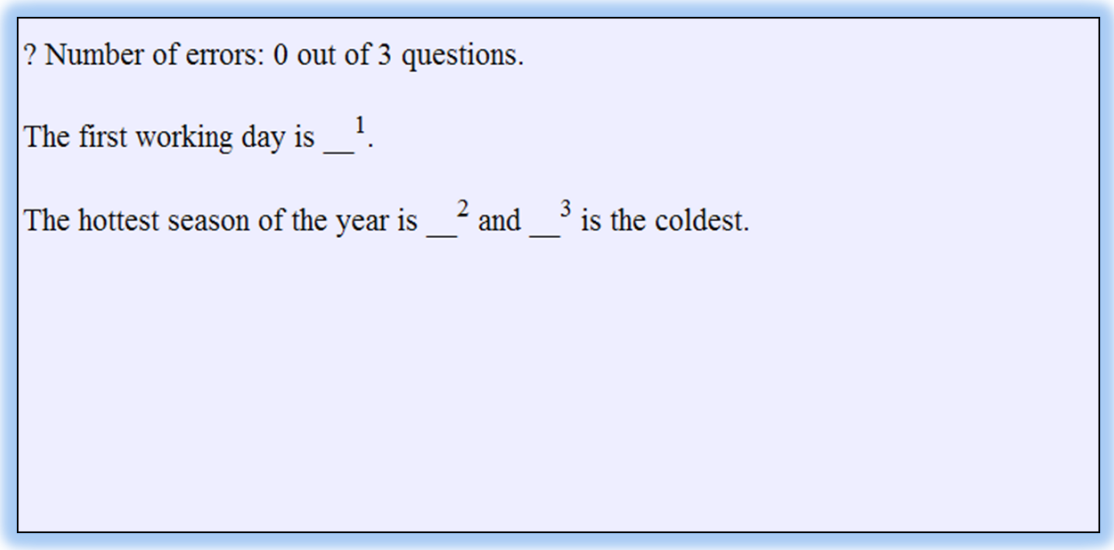

Figure 3.5: Multiple-choice questions

As seen in Figure 3.5 the words Monday, summer and winter are replaced with an underlined blank space. During the quiz, the choices will appear by hovering the mouse over or clicking on the underlined space.

Feedback:

If the correct choice is selected, the drop down menu will disappear and the answer replaces the underlined blank space. On the other hand, if a wrong choice is selected, that incorrect option will turn red indicating that it is a wrong choice. Furthermore, every wrong choice remains red.

Figure 3.6 shows the two examples.

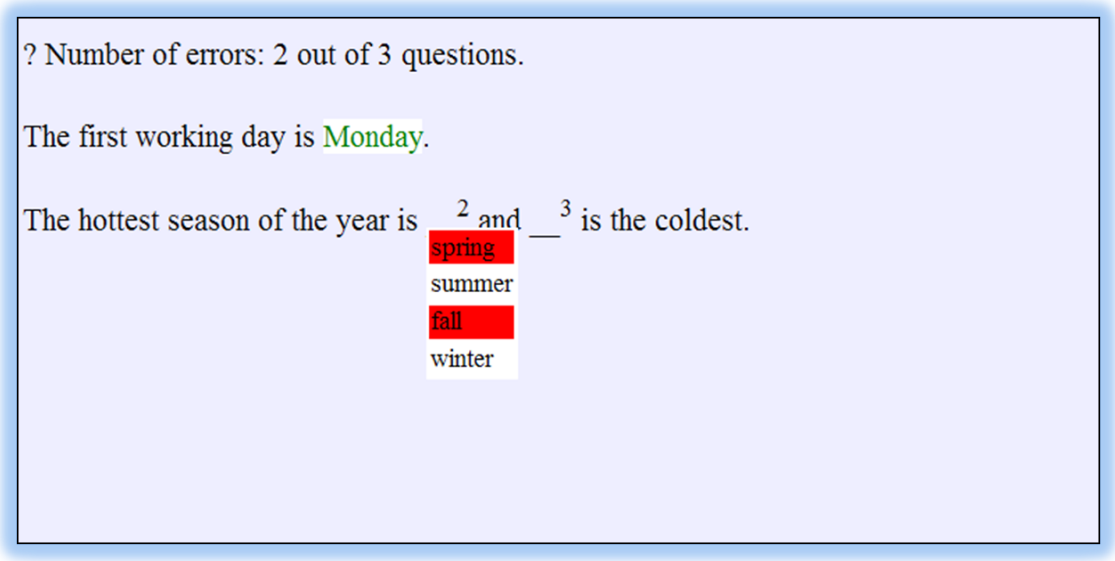

Figure 3.6: Choosing correct and incorrect choices

In-Line Multiple-Choice Questions:

The second method of creating a multiple-choice question is by providing an enumerated list of choices within any sentence. The list starts with the Trigger Symbol "[" and ends with "]". The list of choices is typed within the two brackets and the correct answer is preceded by the Trigger Symbol "@".

Usage:

For example, referring to the "seasons of the year" example in the previous section, one can produce a similar question by writing: The hottest season of the year is [spring @summer fall winter]. The output for the "in-line multiple-choice question" type is the same as that of the "global multiple-choice question" type above and

is shown in Figure 3.5, with the exception that the list of options, embedded in a sentence, can be used in just that particular instance.

Feedback:

Similar to the "global" multiple-choice question type, if the correct choice is selected in the "in-line question", the drop down menu will disappear and the answer replaces the underlined blank space. On the other hand, if a wrong choice is selected, that incorrect option will turn red indicating that it is a wrong choice. Furthermore, every wrong choice remains red.

Clue:

One of the unique functionalities of QuizScript is its ability to provide clues to quiz takers. Instructors can select a word/phrase to act as a "Clue" for the users. The "Clue" is activated when the mouse hovers over it – the word lights up. Once the mouse moves away the "Clue" stays the same colour as the rest of the text. These "Clues" are especially useful for language teachers. The instructor can select a word, which acts as a "Clue", to light up to aid students having difficulty answering the question.

Usage:

Creating a "Clue" in the quiz is as simple as enclosing the selected word/phrase with the Trigger Symbol "*". For example, if the intended question is: "Conjugate the verb: The dog and cat _____ (fight) *yesterday*." One would only type "Conjugate the verb: The dog and cat =fought (fight) *yesterday*." The result is shown in Figure 3.7.

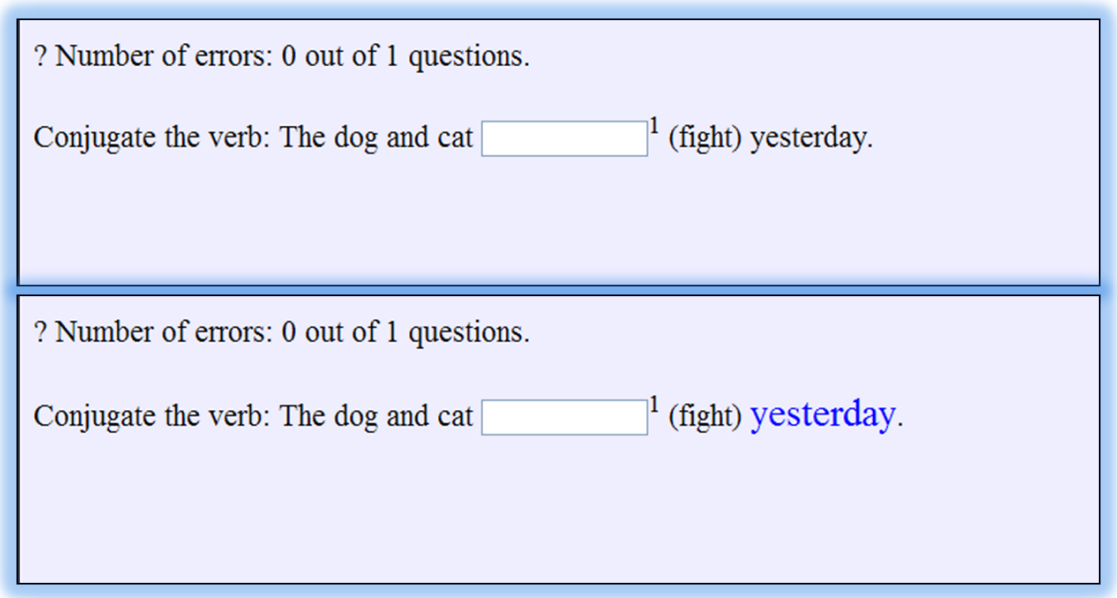

Figure 3.7: Top: The created question. Bottom: Mouse hovered over the "Clue" word

Pop-up Hint:

Another unique functionality of QuizScript is that it has the ability to provide interactive hints. This "Hint" can be used by the instructor to provide a synonym, definition, or explanation of a particular word or phrase. The "Hint" will appear above the selected word/phrase when the mouse hovers over it, otherwise the "Hint" is not shown.

Usage:

To create a "Hint", the instructor encloses both the chosen word/phrase and its explanation ("Hint") within the Trigger Symbols "(" and ")"; the word/phrase and the "Hint" must be separated by the Trigger Symbol "|". The "|" is a bar and not the lower case form of the letter "L".

For example: I am going on a (yacht | a recreational boat or ship) today.

Playing chess is (enjoyable | fun).

Figure 3.8 shows the behaviours of "Hint".

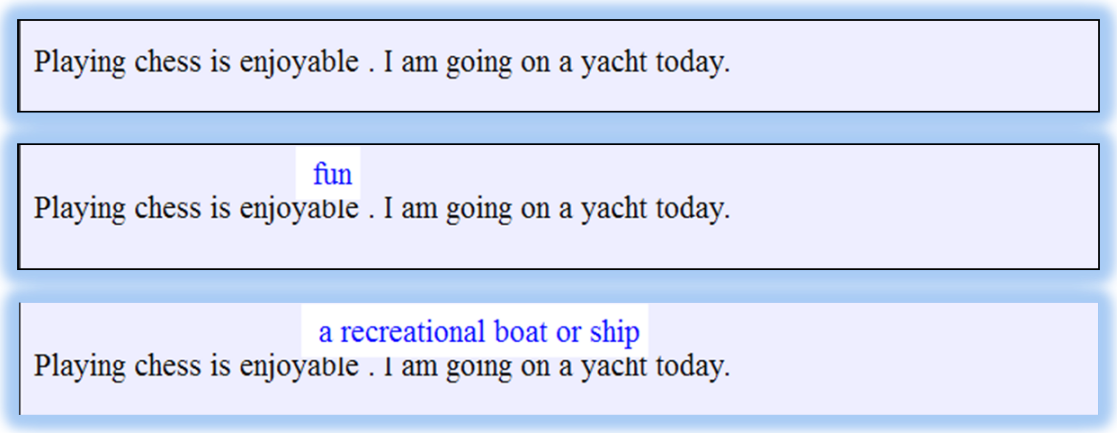

Figure 3.8: Top: Default behaviour of "Hint". Middle and Bottom: Mouse hovered over the words: enjoyable and yacht, respectively

3.1.1 Hidden Features of QuizScript

There are a few other features of QuizScript that are not mentioned in its documentation: 1) Automatic numbering of questions, 2) Terminating characters, 3) Embedded blanks, and 4) HTML insertion.

Automatic Numbering:

While writing a quiz, instructors often want to number each question/sentence/-section. They have the option to manually insert the number themselves. However, later, if they decide to reorganize the order of the questions, then it becomes cumbersome to re-number all of the questions. QuizScript offers an easy solution: type the Trigger Symbol "qq" instead of a number. After submitting the quiz, the system will replace the "qq" with a list of sequential numbers.

Usage:

The Trigger Symbol "qq" can be inserted anywhere in the quiz and QuizScript's engine will automatically replace the symbol with the appropriate number.

For example:

Read the following sentences and answer the questions that follow.

...

Question qq: Where did Mark go?

Question qq: Who did Mark go with?

Question qq: When is Mark going to return?

The above example will output a quiz shown in Figure 3.9:

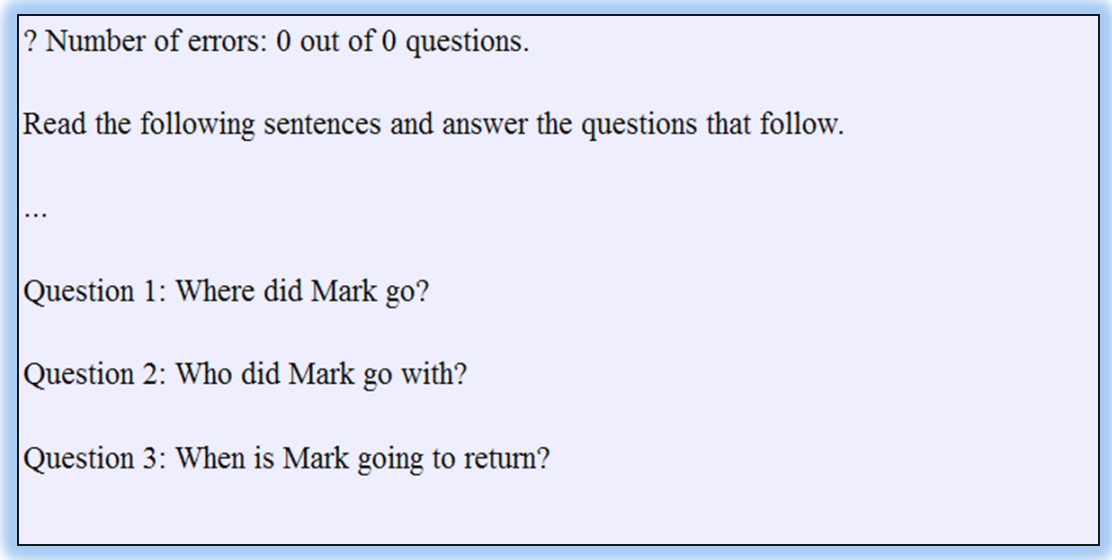

Figure 3.9: Result of automatic numbering using Trigger Symbol "qq"

Terminating Characters: [1]

While creating a multiple-choice or fill-in-the-blank question, the default behaviour of QuizScript is to start interpreting the user input from the Trigger Symbols "#" and "=" and end at any of the Terminating Characters. These characters are: space, period, comma, semicolon, and a new line (pressing the enter key or return key of the keyboard). All but the semicolon character are preserved in the final output in the quiz.

In most cases, instructors want an entire word to be replaced by a blank space, but there are also situations where the instructor wants to insert a blank space somewhere within a word – eliminating one or a few letters only. To accomplish this, one needs to simply insert the Terminating Character ";" after the letter(s) one wants removed. In this case, a part(s) of the word is transformed into a blank space to be answered using a multiple-choice or fill-in-the-blank question. The semicolon is a very useful feature of QuizScript as, unlike the other Ter-

[1]The Terminating Characters are: space, semicolon, comma, and the new-line character.

minating Characters, it does not appear in the final output. This means that a single word will not be interrupted by a punctuation mark which is likely to cause confusion to the reader.

Usage:

This is a very useful feature especially for language instructors. In the French language the same letter may take different accents and change the meaning of the word or its pronunciations. In the following example, referring to the word "prefer" (in French, "préfère"), the instructions request students to select the correct accented letter "e" from the list of choices. The QuizScript's input would be:

.c é è e

pr#é;f#è;r#e;

and Figure 3.10 shows the output while the mouse is hovered over the first blank space.

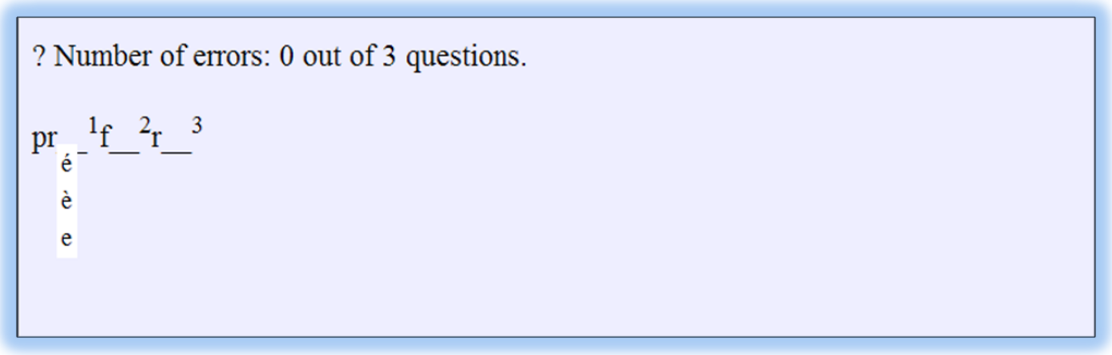

Figure 3.10: Selecting the correct accentuated letter "e" in the word préfère

Embedded blanks

Sometimes, in both multiple-choice and fill-in-the-blank questions, instructors may want to include multiple-word answers. To do this the user needs to link each word in the answer with an underscore. It is important not to include any spaces between the words and the underscore character.

For multiple-choice questions, one generally would list the options for the answer after ".c", and place a "#" before word(s) in the question selected to be turned into a blank space. In cases where the choices consist of numerous words, each word is linked together with an underscore, "_", and each choice is separated

with a space. The following example shows four different choices for an answer: 1) will fight, 2) do fight, 3) are fighting, and 4) fought. As indicated by the "#", the answer to this particular question is "will fight".

List of choices: .c will_fight do_fight are_fighting fought
Question: The cat and dog #will_fight tomorrow.

For fill-in-the-blank questions, the user is required to place an equal sign before the word(s) which are to be turned into a blank space. If an equal sign is placed before three consecutive words, then the output results in three separate blanks spaces.

Example:
If the instructor, however, desires to have only one blank space but a multi-word answer, then they have to do the following: place an equal sign before the first word in the answer, and then link each subsequent word with an underscore. This technique will result in just one blank space within the sentence.
Q: On Monday =the_father_and_the_son went to the mall.

HTML Insertion

Those familiar with HTML may use its capabilities to enhance the quiz by providing links to other websites, add pictures, embed videos or use other HTML functionalities.

3.2 Comparison of QuizScript and Moodle

In this section we will work through an example of creating a quiz using QuizScript. We follow by comparing QuizScript to Moodle regarding their quiz-making process.

Process of quiz-making on QuizScript:

In order to compare the quiz-making process of QuizScript to that of Moodle, we will use the same example as the one used previously for Moodle in section 2.3.1. The example was: suppose an English instructor has just taught a class on Possessive Adjectives (my, your, his, her, its, our, their, whose) and would like to create a quiz composed of a few multiple-choice questions to test or reinforce the students' knowledge on the subject matter.

The desired two multiple-choice questions, and their answers, are:
1) I am writing my book.
2) You are writing your book.

The teacher intends for the underlined words to be replaced with a "blank" allowing the students to choose from the following choices: my, your, his, her, its, our, their, and whose.

After opening QuizScript in a browser, the required steps to generate such a question within a quiz are:

Step 1

Declare the choices to be used throughout the quiz:
.c my your his her its our their whose

Step 2

Type the questions:
I am writing my book.
You are writing your book.

Step 3

Indicate the correct answer in each question by inserting "#" before the word:
I am writing #my book.
You are writing #your book.

Figure 3.11 shows all the required inputs in QuizScript.

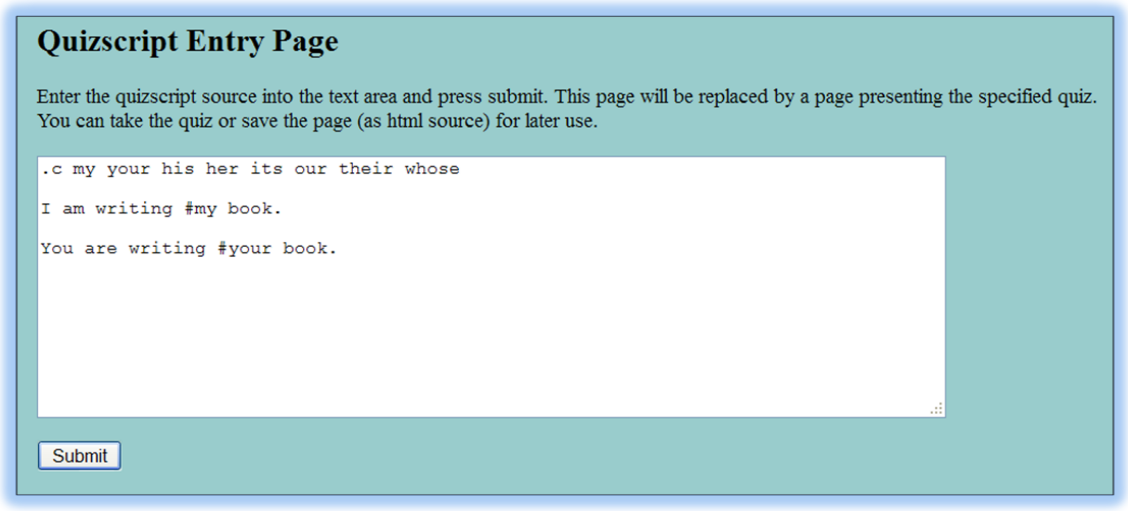

Figure 3.11: The required QuizScript's input to make the quiz in the example

Step 4

Click "Submit" to review and try the quiz, as shown in the following figure:

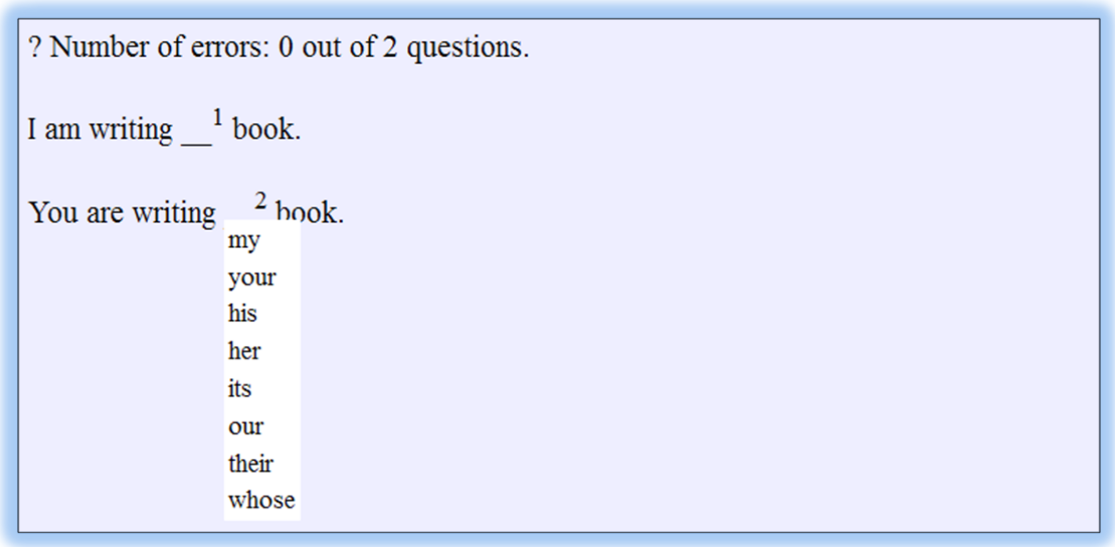

Figure 3.12: The created quiz is shown while the mouse is hovered over the second question

Comparison to Moodle

As shown, an instructor needs to perform only four steps to create the quiz above in QuizScript. To create the same quiz in Moodle, as previously illustrated, requires the instructor to perform at least 20 steps. Moreover, the process we used earlier in Moodle was simplified as we used many of the default settings. In fact, if we had followed the procedures outlined in Moodle's documentation it would have actually resulted in 30 steps to create the same quiz.

Additionally, if more questions are needed to be added then it would take 10 steps per question in Moodle, while it would only take 2 steps per question in QuizScript: 1) type the question; and 2) indicate the correct answer by inserting a "#".

In addition to providing the abilities to create fill-in-the-blanks and multiple-choice questions, QuizScript has 2 extra features that are not available in Moodle. These two features, "Clue" and "Pop-up Hints", are useful pedagogical tools and are unique to QuizScript. Another advantage of QuizScript is that the quiz is interactive and provides immediate feedback to each question. The importance of immediate feedback as a beneficial pedagogical tool is outlined in Chapter 2, where Pressey notes its benefits in quizzes as early as 1920s.

3.3 Draw-backs

Most of the drawbacks in QuizScript are the result of using regular expressions to distinguish the intended questions from the rest of the user input. Each Trigger Symbol is represented by at least one regular expression. In order to create a quiz, one must use the Trigger Symbols and they have to be in the right place and follow the rules governing them. Otherwise, they will not be recognized by the regular expressions.

It is possible that while inserting a Trigger Symbol in the correct place, one may accidentally add another character, a space for example. In this case, the intended question is not recognized by the regular expressions and therefore the question is never created. As an example, if the intended word was to be a fill-in-the-blank and a space was unintentionally added after the equal sign, then the question will not be generated.

Another possibility is that some regular expressions may recognize a token that was not intended to be a question. This is expected as an instructor may use a Trigger Symbol not for the intentions of creating a question, but rather use it for its literal meaning. For example, if the instructor intends to say: "Refer to question #5", then

the character "#" is considered a Trigger Symbol and QuizScript will try to parse it into a multiple-choice question.

Sometimes the Trigger Symbols are used correctly for their intended meanings in QuizScript, but they may not hold the correct or complete information for the question to be completed. This situation can be illustrated with the following example: the instructor uses Trigger Symbol ".c" to declare the choices, and later the Trigger Symbol "#" is used to create the question and inform the QuizScript of the correct answer. However if the chosen answer is not among the list of the choices then the students are presented with the choices but none of them will be correct. In such instances, the system does not detect the error, nor does it inform the user.

In some cases, QuizScript does, indeed, detect errors. For example, if you do not declare your choices in a multiple-choice question, by inserting ".c" followed by the options, then a vague error message will appear. This error message is not useful to the user as it does not indicate the location of the error or a solution to correct it. Only people with a computer science background, and those familiar with the implementation of QuizScript, have the potential to decipher the ambiguous error message. The error message that follows can appear in a variety of circumstances:

Caught MMPException in MMP.main(): Caught exception during macro method invocation: "null"

Another type of error message that may appear is:
Caught MMPException in MMP.main(): Quoted argument lacks closing quote

As mentioned, QuizScript allows for HTML tags insertion. Within the quiz, one can create links to pictures, websites, embed videos, and tables or use other capabilities of HTML. However, HMTL also incorporates some of QuizScript's Trigger Symbols. This can be infuriating as QuizScript will parse part of the HTML tag, which results in an unwanted question and an unrecognizable or uncompleted HTML tag. If the instructor wishes to insert a link in the quiz then the required HTML string would be:

<a href =http://www.somesite.com>click here <\a>. The "=" is a necessary part of the HTML tag but QuizScript's engine will, instead, create a fill-in-the-blank question resulting in both an awkward output and an incomplete link, as shown in figure 3.13:

Figure 3.13: Mistaken HTML tag as a fill-in-the-blank question

An additional frustration is that an instructor's punctuated text, source code, and resulting quiz are on separated pages. This is cumbersome for instructors because, while making the quiz, they have to switch between pages to view their source code and the generated quiz. If one does not like the format, spacing, organization, or word order of the preliminary quiz, then they have to go back-and-forth, blindly, trying to determine the location within their source code to make the appropriate amendment. Beyond these stylistic concerns, addressing the potential punctuation errors, discussed above, is more difficult when switching between pages. Many QuizScript users will be used to "what-you-see-is-what-you-get" type editors, such as Microsoft Word, where the source code and the final product are both immediately present. For these types of users, self-correction would be easier if the source code and a "preview" of the quiz were on the same page — unfortunately, this is not currently the case.

Finally, the created quiz is in the form of an HTML file, which is essentially just a text file, and can be saved on any storage device. QuizScript, however, lacks the means to distribute the quizzes. The QuizScript web-server does not host nor store the quizzes. The quizzes can be copied to a student's storage device, but this method can be very time consuming. With that said, the instructor can upload the quiz to a web-server, providing he or she is aware of the process. Unfortunately, some instructors do not have access to a web-server.

The unique features of QuizScript, which make it easy to generate quizzes, are unfortunately plagued by numerous shortcomings. Improving these drawbacks is the focus of the next chapter, which expands on the solutions that were implemented to the new QuizScript.

Chapter 4

The New QuizScript

The notations used in QuizScript are specifically designed to ease and speed up the process of creating quizzes. Additionally, it offers features that are unique to QuizScript, which are discussed in Chapter 3. However, QuizScript was unintentionally designed with the assumption that everyone would understand the system and not make mistakes. The system can be unforgiving if it comes across a misused Trigger Symbol, as it will not provide any useful information to locate or correct the error. QuizScript also lacks the mechanism to store the quizzes for later retrieval and, subsequently, cannot distribute the quizzes. Even though QuizScript claims to accept HTML tags to be used within a quiz, in reality it does not detect them and often interferes with Trigger Symbols.

During the process, the following array of tools was used to address these problems and implement the solutions. In the back-end of the new QuizScript is a parser programmed using Java platform. More specifically, Java Enterprise Edition (Java EE) is utilized for its ability to create dynamic web-pages. A MySQL database, with its appropriate tables, is created to keep track of and store the quizzes. On the other side, the front-end of the new QuizScript also uses a variety of tools and platforms. The new QuizScript uses HTML to show the content, CSS to provide styling, and JavaScript for interactivities. jQuery, a JavaScript platform, is used for its stability among different browsers. Also, jQuery's AJAX tools are used to bridge the communication between the website and the server. Tomcat Apache web-server is used to host the QuizScript website.

This chapter will focus on the solutions conceived, designed, and implemented to overcome the shortcomings of QuizScript. It starts by providing an overview design of the new system and continues by exploring each of its components in depth.

4.1 Overview of the New System

The new QuizScript consists of numerous components that work together harmoniously. Since the new QuizScript contains many modules, it is necessary to use a software design pattern that separates the business logic from the presentation part of the new QuizScript. "Model-View-Controller" (MVC) software design pattern is used to administer this separation. This design is employed because it allows different components to be manipulated independently, without affecting the other components. For example, if at a later time the layout of the website needs to be altered changes only have to occur in one location – the View Layer. If the database, as another example, needs to be changed then only the Model Layer is altered leaving the rest untouched.

Figure 4.1 summarizes the MVC design used in the new QuizScript.

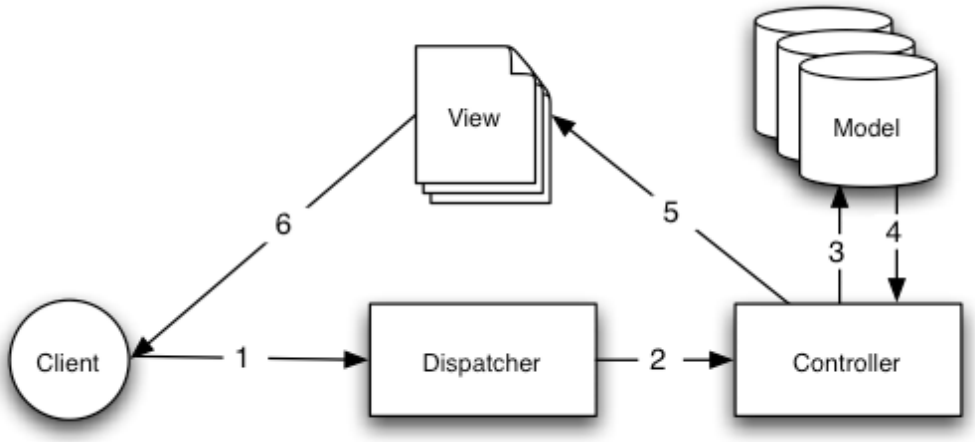

Figure 4.1: Model-View-Controller (MVC) design used in QuizScript

The "Client" is any person that uses a web-browser to connect to QuizScript's web-server and uses its services. Except for some trivial tasks, such as fetching the CSS and JavaScript files that are automatically handled by the web-server, all the requests are first received by the "Dispatcher". The Dispatcher is responsible to direct the client's request to the proper module in the "Controller Layer". The Controller is responsible to talk to the "Model Layer" to get the required data, either from the database or the parser, and transfer them to the "View Layer". The Model Layer is the back-bone of QuizScript and is responsible for handling the data. The parser to

analyze the quiz and the database to store the quizzes are in this layer. Finally, the View Layer is responsible for creating a user-friendly-visual presentation of the data: HTML pages.

Different layers of the MVC are explained in detail in the following sections. We begin with a look at the Client and the Dispatcher, followed by the Controller Layer and the Model Layer, ending with the View Layer.

4.2 Client and Dispatcher Layer

The client will use a web-browser to connect to QuizScript's web-server to access the services offered by QuizScript. The interactions between the client and QuizScript are called "request" and "response". Requests are the messages and the data that are sent to QuizScript, such as requesting a quiz, or verifying its validity. Responses are the messages and the data that QuizScript provides to the user, such as the HTML file that contains the quiz.

There are two types of requests. Some requests are automatically sent by the web-browsers that the user cannot control. These types of requests are automatically handled by the web-server, such as fetching the JavaScript, CSS, and image files. The second type of request are those that the client will explicitly send to QuizScript, such as sending a quiz to the server to be stored in a database. These types of requests are first handled by the Dispatcher.

The complete implementation of the Dispatcher is provided in Appendix D. The underlying process of the Dispatcher is to parse the received URL and determine the proper module from the Controller Layer that can handle the request. For example, if the client needs to view a specific quiz in the database, the URL http://dbweb.cs.uvic.ca:8080/quizscript2/quiz?id=1 is sent to the web-server. After receiving the request, the Dispatcher will parse the URL. The Dispatcher will then pass the request to the module in the Controller Layer that is responsible for fetching the requested quiz.

One may question why not send the request directly to the Controller, bypassing the Dispatcher altogether? The Controller can, in fact, do the job of the Dispatcher, however, it will address each request sequentially. Whereas, by introducing a Dispatcher, multiple requests can be handled concurrently. Therefore, by separating the two layers, the Dispatcher can be used as a load balancer. Many clients can use the system at the same time, and the web-server can create a thread response to each

request separately. The Dispatcher will quickly detect the users' needs and forward the requests to the proper module in the Controller. This way the Dispatcher acts as a load balancer and many different Controller modules can be working at the same time. Additionally, the Dispatcher can be utilized to do the repetitive work required for every request, such as ensuring the user is logged-in.

4.3 Controller Layer

The modules in the Controller Layer are responsible for rendering the client's requests and taking the appropriate actions. They communicate with the modules in the Model layer to retrieve or store the quizzes and invoke the Parser to check the validity of a quiz. The results of the action is forwarded to the View layer to be visually shown to the client.

The complete implementation of the modules that make up the Controller Layer are given in Appendix B and their explanation follows:

HomePageServlet

QuizScript's home page shows the 5 most recent quizzes added to the database. The HomePageServlet will retrieve the quizzes and pass them to the module in the View layer responsible for displaying the home page. Even though by default it retrieves the 5 most recent quizzes the user can choose other values.

Parser

This class is responsible for invoking the Parser module in the Model layer to check for the validity of the in-putted quiz. If the quiz is formed correctly then the created quiz is sent to the View layer otherwise an appropriate error message is sent.

Quiz

Retrieves a specific quiz's source code from the database, creates the quiz, and forwards it to the View layer.

SaveQuiz

Renders the required information needed to save a quiz in the database from the user and passes them to the Model layer to be saved.

WadgeMMP

Since the new QuizScript is a work based on the QuizScript created by the Wadges, this class is the bridge between the two. Once the in-putted quiz is checked for correctness it is passed to the MMP engine in Wadges' QuizScript. In addition to the quiz's source an MMP file called "yasser-quiz.mmp" is also passed to the MMP engine. The resulted quiz is created and captured by this class to be sent to the View layer and shown to the user.

UTF-corrector

In order for QuizScript to support a variety of languages, UTF was used. The character encoding for the web-pages and the communication between the modules needed to be set at the earliest point of receiving a message. Every connection and their subsequent interaction is first handled by this class and it sets the proper character encoding.

4.4 Model Layer

The business logic part of the system is implemented in the Model Layer. The modules in this layer receive requests from the Controller Layer and process data accordingly. There are two main components that make up this layer, the parser, which will be discussed in depth in the next chapter, and the database, discussed below.

In order to host and store the quizzes on the new QuizScript's website, a database is created. The MySQL database stores the quizzes and the related information about them. The quizzes are automatically indexed by Database Management System (DBMS), therefore optimizing their retrieval times. Databases also support multi-threaded and concurrent access to the data while being watchful of data integrity. The use of a database allows for future changes to QuizScript to be easily implemented. If at a later time the need arises to keep track of users or to require them to log into the site, the required tables can be generated in the database creating a relationship between the quizzes and the users.

Retrieving data or accessing it from the database requires making a connection with the DBMS. Since QuizScript's website is dynamic it needs to interact with the database, sometimes per request. Creating a connection to the database for each request is expensive and time consuming. The solution found to overcome this is to use a "connection pool". In a connection pool scheme, multiple connections are

established at the system start-up. During the life time of the program if a connection needs to be made to the database, a connection that has already been created is used. Once the need for the connection vanishes, it is closed. Closing a connection does not actually terminate the connection, rather it is simply returned to the connection pool for future use.

Apache web-server provides connection-pool-management tools for QuizScript [22]. An XML file called context.xml is created to configure its database-connection-pool mechanism. The fully documented context.xml file can be found in Appendix E.2 and contains a Resource tag with the following attributes:

```
1  <Resource
2    name="jdbc/TestDB"
3    auth="Container"
4    type="javax.sql.DataSource"
5    maxActive="100"
6    maxIdle="30"
7    maxWait="10000"
8    username="intentionally not provided for security reasons"
9    password="intentionally not provided for security reasons"
10   driverClassName="com.mysql.jdbc.Driver"
11   url="jdbc:mysql://127.0.0.1:3306/quizscript?useUnicode=true"
12 />
```

All the required information to connect to the database, such as its URL, username, password, and number connections, is configurable from this file. This approach provides another level of abstraction, as any information related to connecting to the DBMS is included in only one place.

Upon successfully configuring the context.xml file, the file web.xml in WEB-INF folder needs to be modified to include:

```
1   <resource-ref>
2       <description>DB Connection</description>
3       <res-ref-name>jdbc/TestDB</res-ref-name>
4       <res-type>javax.sql.DataSource</res-type>
5       <res-auth>Container</res-auth>
6   </resource-ref>
```

The complete web.xml file can be found in E.1.

Within QuizScript's database, the most important table is the quizzes table. This table holds the records of all the quizzes in the system and includes the following attributes:

ID

A unique integer created automatically by the DBMS that is used to index the quizzes. Optionally, since its value is sequentially incremented, the last value of ID corresponds to the number of quizzes in the database.

Date

The date that the quiz was entered in the database. Its format is YYYY-MM-DD (year-month-day).

Description

A field that allows for a short description of the quiz, up to a maximum of 100 characters.

ViewCount

Holds the number of times a specific quiz has been retrieved for viewing purposes.

Source

This field holds the punctuated text created by an instructor. It corresponds to the source code of the quiz, which QuizScript can turn into an HTML file – the outputted quiz – at a later time.

4.5 View Layer

Upon visiting the new QuizScript's website, the user is shown the homepage, as shown in Figure 4.2.

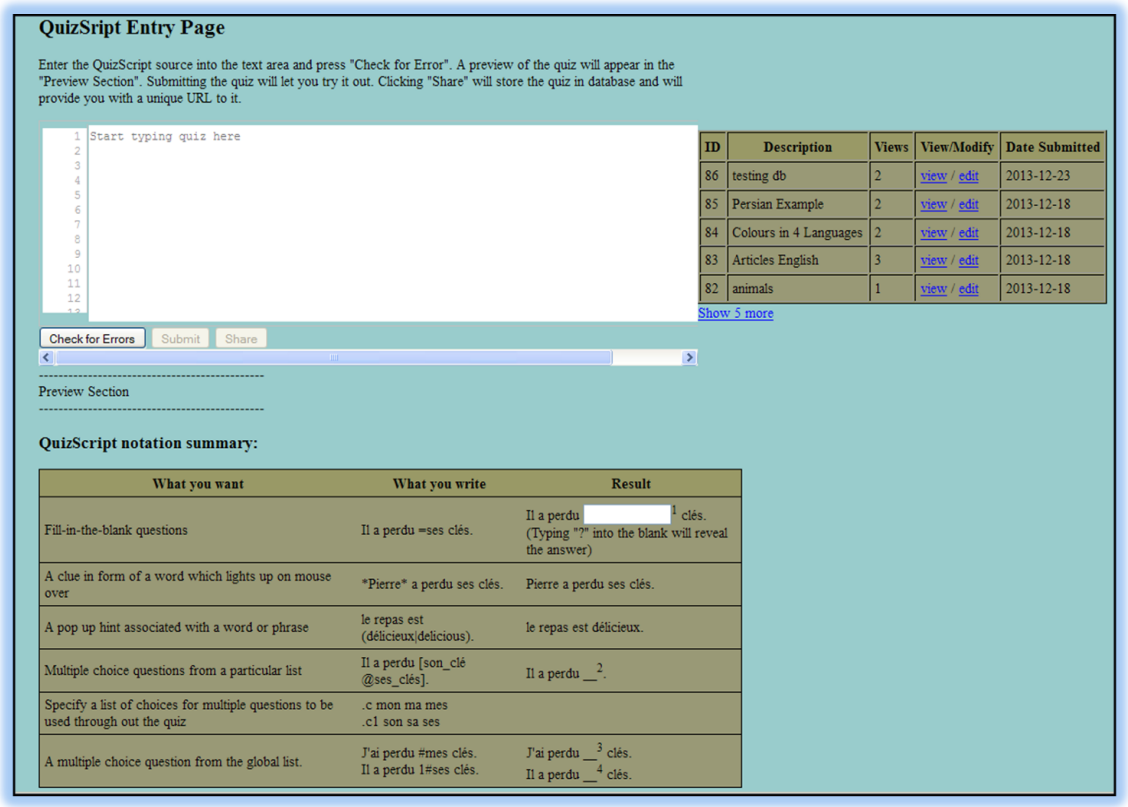

Figure 4.2: QuizScript's home page

The homepage consists of 3 sections:

Text Box

The quiz and its marked-up characters are typed in this section to be checked by the Parser.

Preview Section

The area underneath the text box is called the Preview Section. The in-putted quiz is checked for correctness and if there are any errors it will be shown in this section. If the quiz does not contain any errors then a preview of the quiz is shown instead.

Recent Quizzes

The right side of the screen contains the list of the 5 most recent quizzes added in the system. One can view such quiz or edit it, which leads to the creation of a new quiz.

The functionalities and features of the new QuizScript will be explained in following sections using the same example as used in Chapters 3 and 4.

4.5.1 Immediate Preview

The new QuizScript allows users to immediately view the quiz while still working on it. Changes can be made to the quiz and the results are shown on the same page directly below the quiz-in-progress. Clicking "Check for Errors" button will verify the correctness of the input and if there are no errors then a fully functional quiz is immediately shown. Fig 4.3 illustrates the preview of a partially completed quiz.

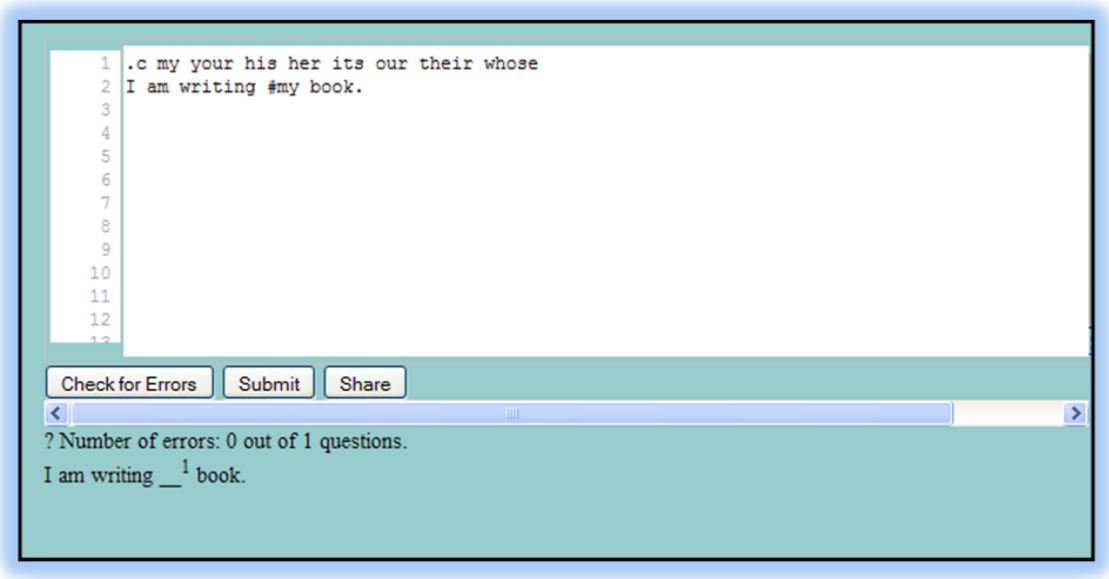

Figure 4.3: Preview of a partially completed quiz

4.5.2 Useful Error Messages

If the in-putted quiz contains errors the Parser will provide useful error messages that can be used to easily identify the problem. Additionally, the line number containing the error is highlighted in red to further help with detecting the error.

Figure 4.4 illustrates the feedback received by the user while creating a quiz containing an error. In this case the word "my" was mistyped as "mye" which is not among one the choices initially given.

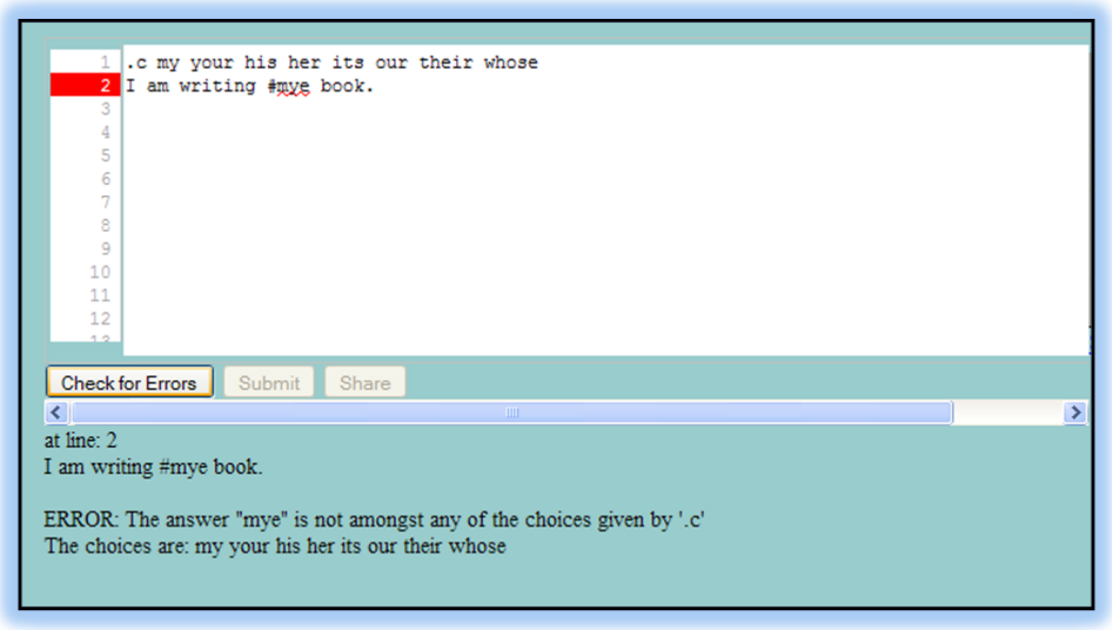

Figure 4.4: Error notification

4.5.3 Stand-alone Quiz

A correctly formed quiz can be viewed by itself without the other components on the site. Clicking the "Submit" button will create an HTML file containing all the required JavaScript. The resulting page, as shown in Figure 3.12 , and as previously mentioned, can run by itself in a web-browser. It can be saved and later distributed as an HTML file, which is essentially a text file.

4.5.4 Storing the Quiz

The error-free quiz can be saved in the database. Once saved in the database it will be shown in the list of recently added quizzes to the system and can be altered or viewed by anybody accessing the site. Additionally, a unique URL is created for the quiz which can be used to directly access the quiz.

Saving the quiz to the database is as simple as clicking the "share" button and filling-in the required fields on the screen. Figure 4.5 shows the pop-up screen requiring the author's name and a short description of the quiz after clicking the button.

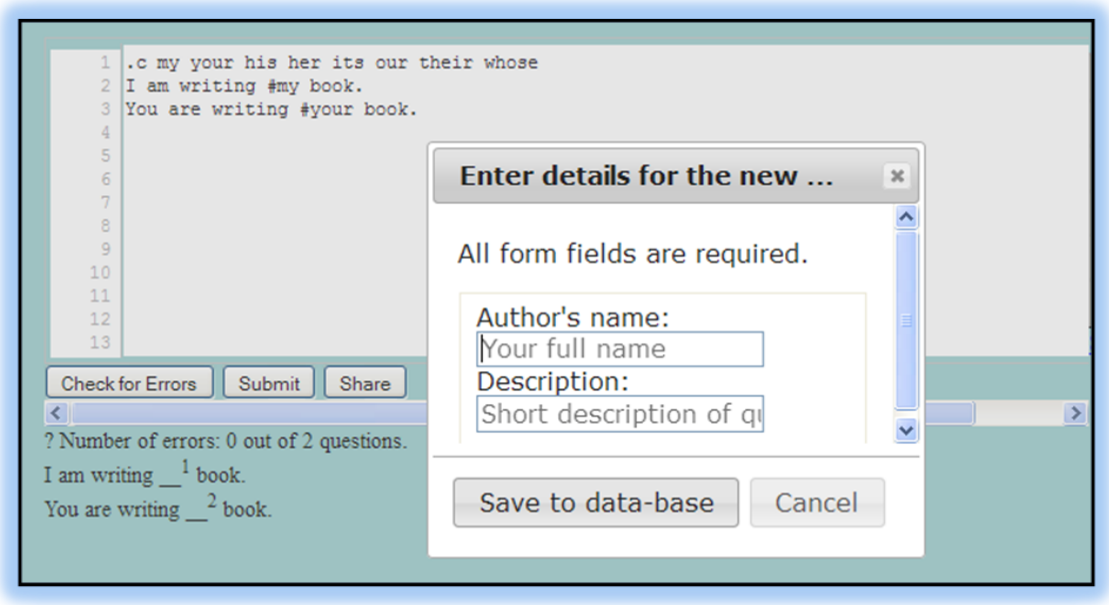

Figure 4.5: Required information to save the quiz in database

A unique URL containing the exact address of the saved quiz is provided in the preview section. This URL can be used by any web-browser to access that specific quiz. The URL is in form of http://dbweb.cs.uvic.ca:8080/quizscript2/quiz?id=1 where the number at the end of the URL is the quiz's unique ID.

The following chapter outlines the Parser that was implemented. The Parser is used to validate the correctness of the input and provide useful information to detect and correct the errors.

Chapter 5

Parser

A large problem with QuizScript is its lack of error detection. As a result, a new parser had to be designed and implemented that would detect the errors and, more importantly, provide a useful description of them. This chapter outlines a detailed discussion of the Parser that was implemented.

QuizScript uses regular expressions as its method of parsing the punctuated text into a quiz. However, regular expressions either find a token or do not. This method does not provide any feedback to the user indicating whether a question was incorrectly formed or missing integral aspects. A more improved approach is by taking QuizScript out of the constraints of regular expressions and implementing a recursive descent parser with Deterministic Finite Automata (DFA) functionalities.

In order to detect the errors, the parser needs to know how to identify an error. Therefore, we defined a "language" and the syntax for QuizScript. A language consists of a finite alphabet and a finite set of rules [23]. Based on the rules of the language, the parser can then recognize whether the input: 1) is correctly constructed, in that it follows the rules of the language, or 2) has an error, pushing the parser to send a detailed error message to the user[1]. A "top-down recursive descent" [23, 24] approach was used to design the parser. A top-down parser checks to see if a string can be generated by the grammar starting from the initial symbol working its way down the input. Since the parser is composed of mutual functions that descend through the input, it is referred to as recursive descent [24]. More specifically, our recursive parser is in the class of LL(1) parsers: it reads from left-to-right allowing only a one-symbol lookahead.

[1]A detail explanation of this set of rules will be explained later in section 5.1

Despite there being other possible methods, we chose to use the recursive descent approach because of its ease to implement. In our recursive descent model, the parser calls the appropriate function when it encounters any Trigger Symbol. Essentially, every Trigger Symbol has its own function, each breaking up the code into logical segments. Recursive descent also allows for future changes to be easily made to QuizScript's language. At a later time, if the need arises to add Trigger Symbols then implementing its corresponding function, leaving the rest of parser unchanged, is all that needs to be completed.

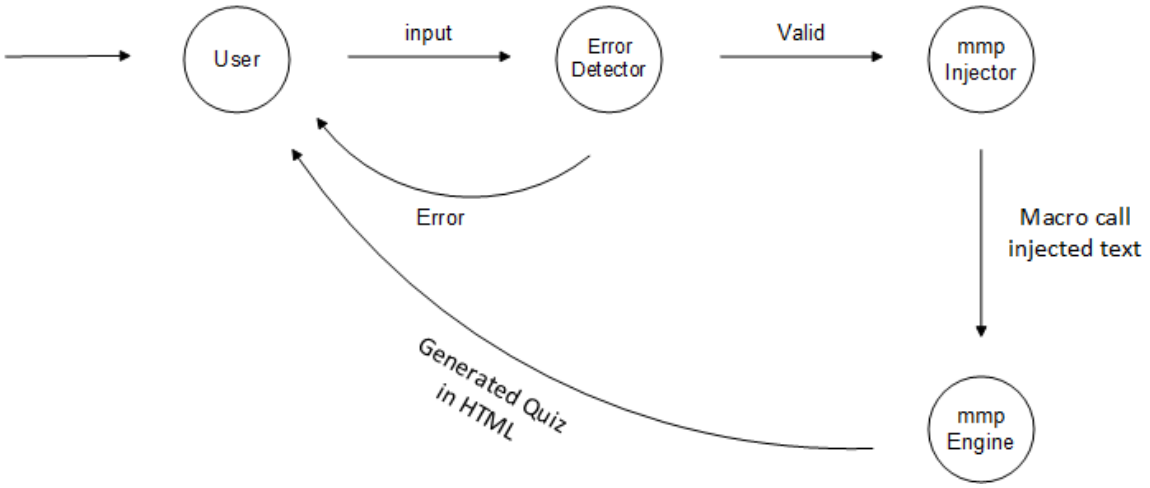

Figure 5.1: The work flow of the new QuizScript

Upon receiving the user's input, the parser will scan the input twice. First, the parser passes the input to its "Error Detection" mechanism, and if no errors are found the input will then be scanned by the "MMP Injector".

5.1 Phase 1: Error Detection

A user can make numerous types of mistakes anywhere while punctuating the text. It is therefore important for the parser to know what can go wrong at each stage so it can clearly describe the error to the user, allowing them to correct the input. To accomplish this, we used the DFA process. Using the DFA, the parser would be in a particular state after reading the input. Based on that state, the parser can then know what it has parsed thus far, what it needs in order to transition to the next state, and, given any state, what characters would cause an error. Since the states

are transitioned based on the characters provided by the user, the parser can then recognize which characters are valid at each state and which characters are not. This method will create a clear error message indicating what caused the exact problem thus, allowing the user to easily correct the error in their input.

The alphabet used in the language is all the characters defined by UTF-8 – essentially, all the characters in every language. The syntax for the grammar, in EBNF format, and their corresponding DFA are individually explained:

Fill-in-the-Blank Question:

Syntax:

```
1 <fill-in-the-blank> := "=" <word> <terminating characters>
2 <word> := <character>+
3 <character> := any character of UTF-8
4 <terminating characters> := . | , | ; | space character | new line
```

States and Transitions:

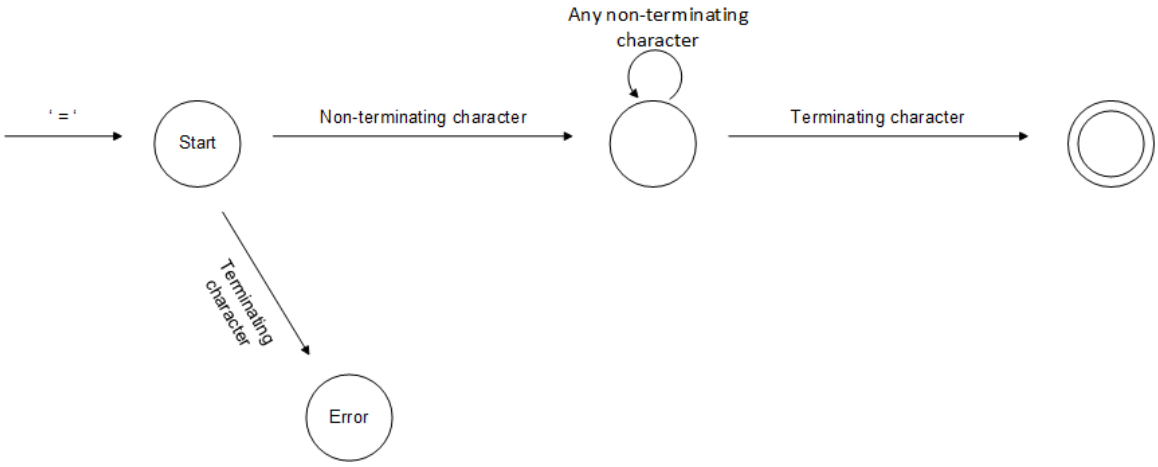

Figure 5.2: Fill-in-the-Blank DFA diagram

When the parser sees an "=" it will use the DFA above. Given that it has read the first character, it will then start to read each subsequent character one at a time, until it sees a Terminating Character issuing it to stop. Along the way, it will also recognize characters that are invalid.

Global Multiple-Choice Questions:

Syntax:

```
1 <global multiple-choice> := <digit>? "#" <not allowed characters> <
     answer> <terminator>
2 <digit> := utf8 digits
3 <space> := space character
4 <answer> := <utf8char>+
5 <terminator> := <space> | . | , | ; | \n
```

States and Transitions:

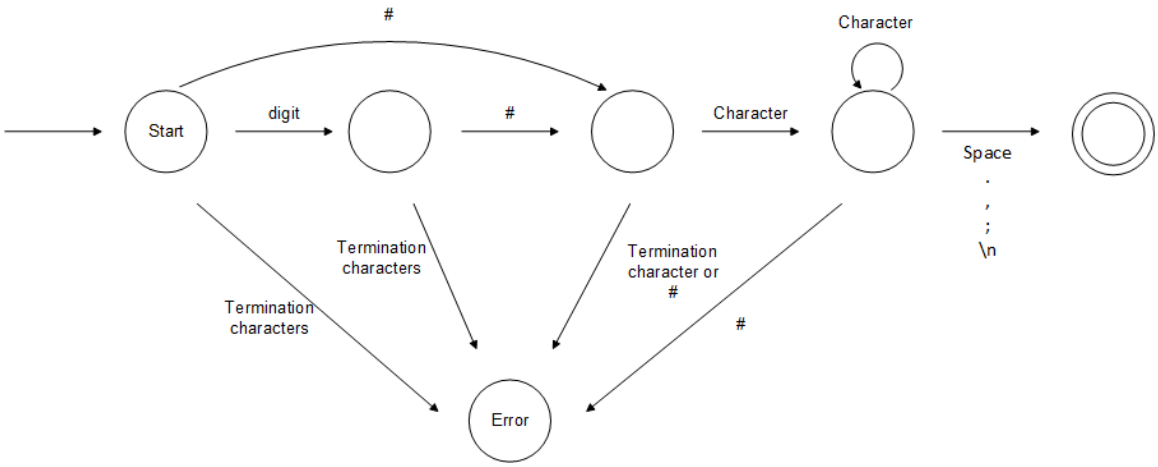

Figure 5.3: Global multiple-choice question DFA diagram

The "#" indicates the correct answer to be used for a multiple-choice type question. In the case where there is more than one list of choices, each list is distinguished by a digit. In the syntax, this particular digit immediately follows "#". If the indicated answer is not among the list of choices given by Trigger Symbol ".c" then an error is detected. The error message tells the user what the possible choices are so they can fix the problem.

Declaring a List of Choices for Global Multiple-Choice Questions:

Syntax:

```
1 <list of choices> := ".c" <digit>? "space" <choices> <new line>
2 <digit> := any utf-8 digit
3 <choices> := <word>+
4 <word> := <utf8char> | <utf8char> <space>
```

States and Transitions:

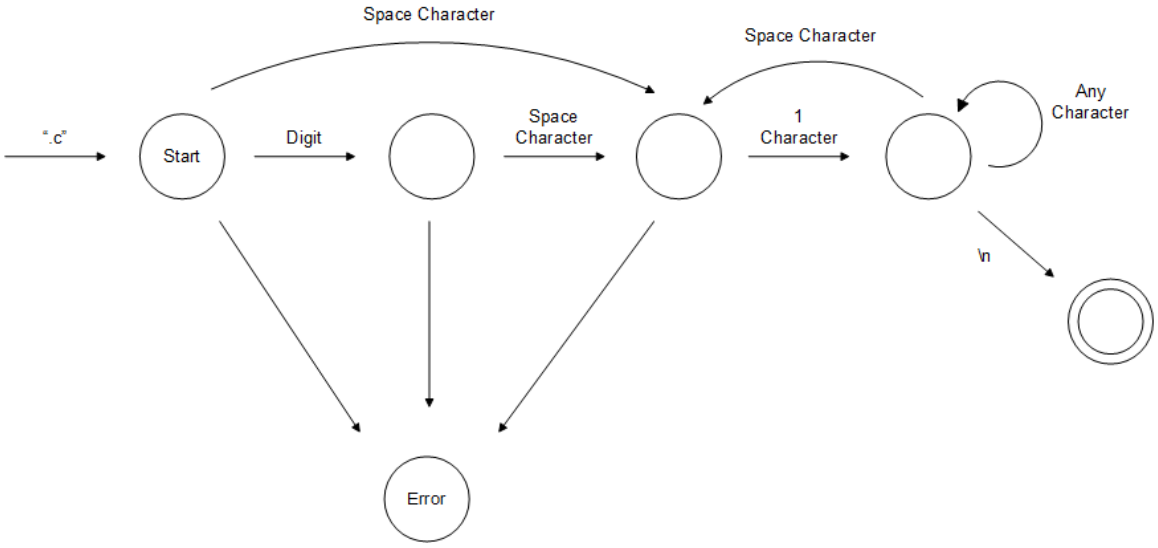

Figure 5.4: DFA diagram for declaring a list of choices for global multiple-choice question

The list of choices throughout the quiz for multiple-choice questions starts with ".c" and is then followed by the choices, which are space separated ending with a new line. In the case where there is more than one list of choices, each list is distinguished by a digit immediately following ".c".

In-Line Multiple-Choice Question

Syntax:

```
1 <inline multiple-choice> := "[" <question> "]"
2 <question> := <choices>* <answer> <choices>*
3 <answer> := "@" <word>
4 <word> := <utf-8 except "[" and "@">+ <space>
5 <space> := " "
6 <choices> := <utf-8>+ except <space> or "@"
```

States and Transitions:

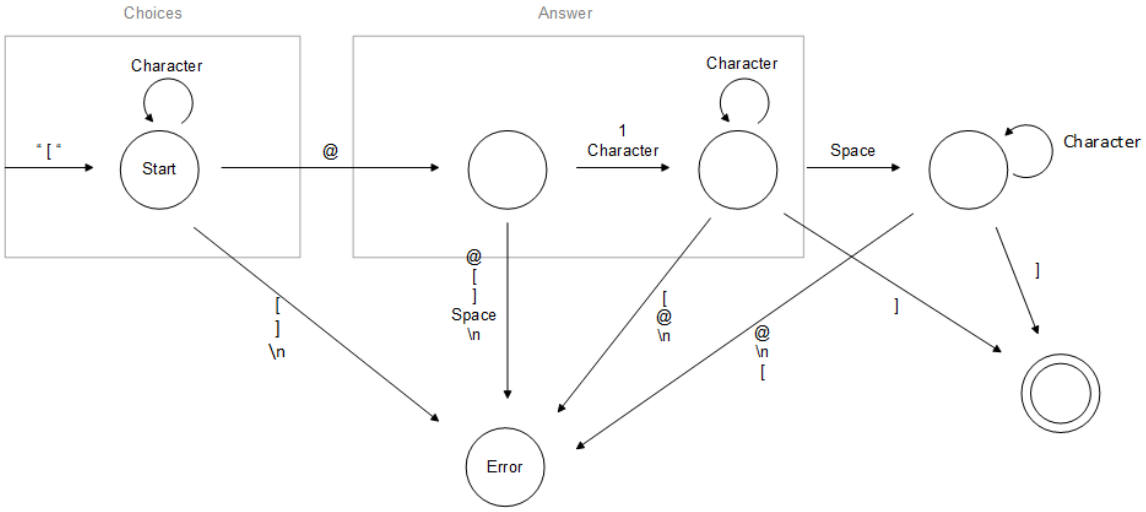

Figure 5.5: DFA diagrams for the in-line multiple-choice questions

A list of choices housed by "[" and "]" where the correct answer is indicated by "@".

Clue

Syntax:

```
1 <clue> := * <word> *
2 <word> := anything more than one character except space or newline
```

States and Transitions:

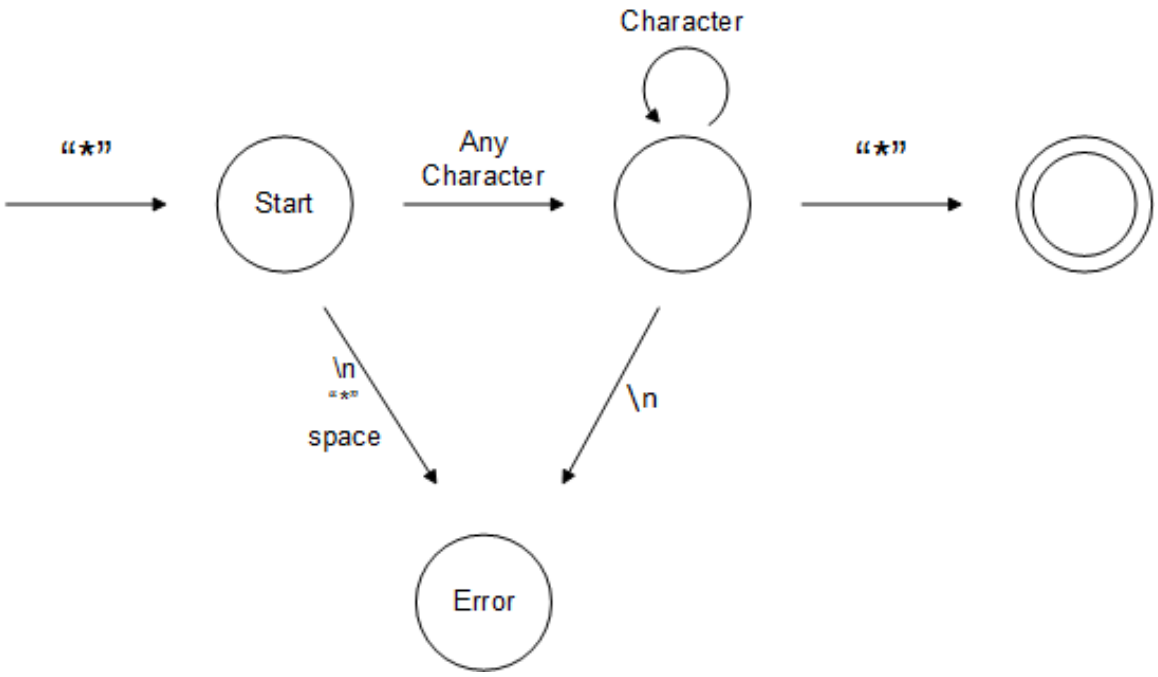

Figure 5.6: DFA diagrams for clues

A “Clue” is identified by a word wrapped with “*”. The word can be of any length. Since only one word is allowed per clue, we do not allow spaces or new lines as valid characters for this case.

Pop-Up Hint

Syntax:

```
1 <pop-up hint> := "(" <words> "|" <hint> ")"
2 <words> := anything more than one character except (, |, ), or \n
3 <hint> := <words>
```

States and Transitions:

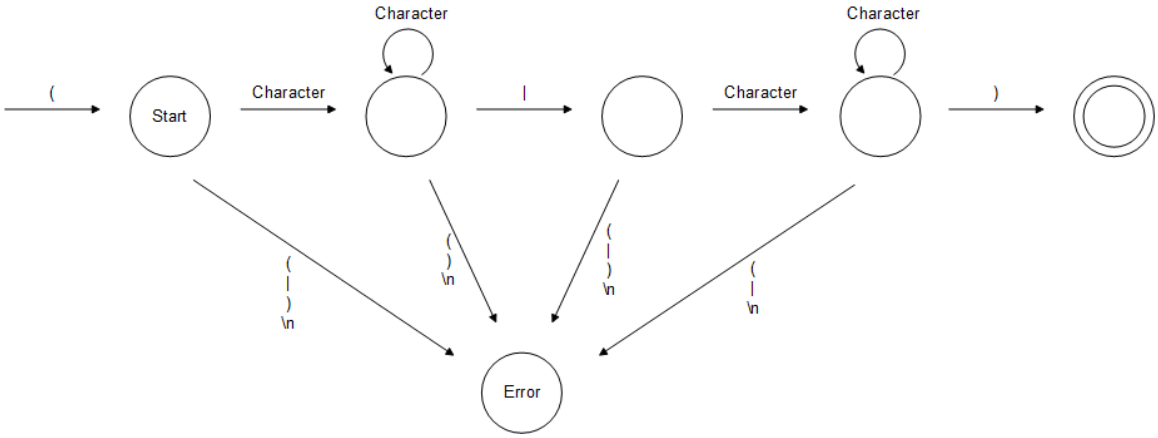

Figure 5.7: Pop-up hint DFA diagram

Two phrases of any length are separated by a bar and both phrases are enclosed by parenthesis. We do not allow nested hints therefore the use of parenthesis is only allowed on the outer edges of the strings. Since the parser reads one line at time, we do not allow the pop-up hint to be spanned over multiple lines.

5.2 Useful Error Messages

Upon detection an error in the input, it is important that the parser provides enough information to the user to easily detect and correct the error.

Since we use DFA to parse the input, we are able to identify at which state an error is created. The error messages returned to the user include the line number, the type of question/feature they were trying to make, the specific error message, and a suggestion as to how to fix the error. Moreover, the line containing the error is clearly highlighted in the web-browser to further help identify the mistake.

For example: if the intended input is, "The hottest season of the year is [spring @summer fall winter]", and the instructor forgot to indicate the answer by using the Trigger Symbol "@", the sample error message would be:
Error at line 1:
>>>The hottest season of the year is [spring summer fall winter].
In-line-multiple-choice question.
The answer was not found.
Insert "@" before the correct choice.

The same example is illustrated in Figure 5.8.

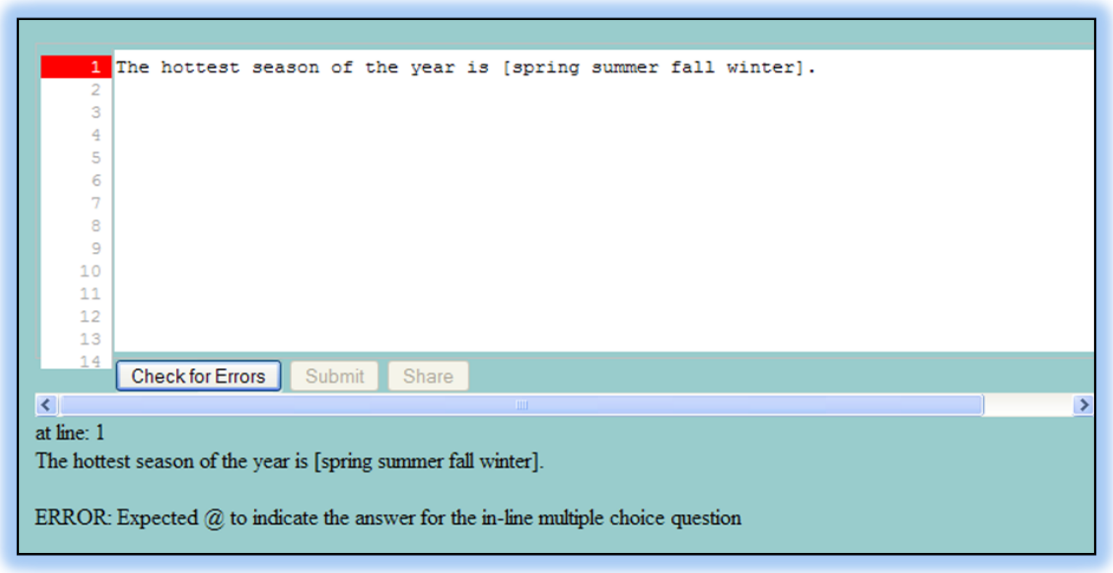

Figure 5.8: An example of an error message

If an error is present in the quiz, the parser will stop and create a proper error message, which it then sends to the user. The remainder of the input will not be parsed until the error is solved. After all the errors have been corrected, the error detection part of the parser is complete and it will continue on to Phase 2 – MMP Injector.

5.3 Phase 2: MMP Injector

The user's input is parsed once more at this stage. Similar to Phase 1, the parser uses the same DFA. During this stage it can be safely assumed that there are no errors in the input, as they were caught by the Error Detector functionality of the parser.

Upon detecting a Trigger Symbol, the parser will scan the input until it encounters a terminating character. At this point the punctuated text, starting from Trigger Symbol to the Terminating Symbol, is replaced with its appropriate macro call to be evaluated later by QuizScript's engine.
The macro calls are described in Table 5.1.

Table 5.1: The macro calls used in QuizScript

Question Type	Macro Call	Description
Fill-in-the-Blank	{tr $1}	$1 is a placeholder for the word to be replaced by a blank.
Global Multiple-Choice Question	{pq_ $2 $1} {pq_ $1}	$2 is replaced by the number preceded by ".c", and $1 is a placeholder for the word to be replaced by a blank. If ".c" did not include a digit, then the macro call {pq_ $1} is used instead.
Declaring List of Choices for Global Multiple-Choice Question	{choicesi $1 $2} {choices $1}	$1 is replaced by the digit, and $2 is replaced by the list of choices. {choices $1} is the macro call used if ".c" was not followed by a digit, and $1 is replaced by the list of choices.
In-line Multiple-Choice Question	{cq $1}	Everything enclosed by "[" and "]" replaces $1.
Clue	{bl $1}	Everything enclosed by "*" replaces $1.
Pop-up Hint	{tt $1 $2}	Everything from "(" until "|" takes the place of $1 and from "|" until ")" takes the place of $2.

5.4 Escape Character

Trigger Symbols themselves can be problematic as they also function as punctuation marks and general typography. There will be cases where an instructor requires the use of one of these "symbols" in their text as per their intended literary function, rather than as a Trigger Symbol to create a question in a quiz. An example of this is embedding an article into a quiz to serve as the preamble to a set of questions. To address this problem, another character was added to the list of Trigger Symbols, "\". This allows one to instruct the parser to omit any Trigger Symbol following the "\", and not to treat it as a typical Trigger Symbol.

Example: “1 + 2 \=3” will not turn into a question but rather it will be shown as “1 + 2 =3”.

“1 + 2 =3” results in → 1 + 2 ___
“1 + 2 \=3” results in → 1 + 2 = 3.

5.5 HTML Handling

QuizScript allows instructors to insert HTML inside a quiz. Sometimes the HTML tags contain the same characters as the Trigger Symbol used in QuizScript. Therefore we implemented HTML tag detection in the parser directing it to skip over the tags. Since all of the HTML tags are enclosed within “<” and “>”, the parser now knows that it should stop parsing when it sees “<” and should resume parsing immediately after “>”. However, the “<” may have been incorporated into the quiz for its typographic function, in cases such as: “1 < 2”. In this case, if the parser cannot find the closing “>” it will return to the opening “<” and continue parsing immediately after it.

The following chapter will conclude the thesis and provide a summary of the shortcomings of the original QuizScript, and the methods used to overcome them.

Chapter 6

Conclusions

In this thesis, we explored and implemented a top-down recursive parsing method for QuizScript to overcome its drawbacks. New functionalities were added to allow instructors to skip over Trigger Symbols, and to provide them with useful messages to detect and correct errors. The parser was designed to detect and skip over HTML tags so they would not interfere with QuizScript's engine. A dynamic website, in conjunction with a database, was developed to store, host, and view the quizzes. The website also allows for the immediate preview of a quiz while it is being created, which is another new feature of QuizScript.

To contextualize QuizScript, in the beginning we looked at the history of LMS as early as 1924 and examined Moodle as an example. We focused on the process of making quizzes on Moodle to show how a tedious number of tasks is required of the user to make a simple quiz. This frustration was what initially motivated and inspired the Wadges to create QuizScript.

This thesis includes a detailed explanation of the Wadges' QuizScript and the process of how to create quizzes using this system. Even though Wadges' QuizScript simplified the process of making quizzes, it contained a number of glaring obstacles. To overcome these shortcomings a new QuizScript was created.

The details of the new QuizScript thoroughly explains how it solves the problems of its predecessor. The new website allows for immediate feedback and also solves the distribution limitation. A parser was implemented to verify the correctness of the quiz and provide useful information to the user to easily detect and correct the errors. The improvements make this pedagogical tool, QuizScript, extremely beneficial to both instructors and students, and its uses are far reaching from all levels of education to training in the corporate sector.

Appendix A

Implementation of Model Layer

The Parser's module responsible for checking the validity of the input is given in Appendix A.1 and the MMP Injector class is in Appendix A.2. The classes responsible for communicating with the database are in Appendix A.3

A.1 Parser Error Checker Implementation

```
1  package modelLayer.parser;
2
3  import java.util.regex.Matcher;
4  import java.util.regex.Pattern;
5
6  public class QuizScriptParser {
7
8     private String qsource;/* = "";*/ //the quizscript source file to be
          checked.
9     private int cursor = 0;       //a pointer to where in the qsource is
          being read.
10    private int marker = 0;  //used to mark certain parts of code to show
          what they are.
11    private boolean errInWord = false; //used to show error in fill_in_blank
           method
12    private int numChoices = 0; //used to show number of choices in the
          global multiple choice section.
13    private boolean error = false;
```

```
14    private String errorMsg;/* = "";*/
15    private String[] globalMultChoices;
16
17    public QuizScriptParser() {
18       errorMsg = "";
19       globalMultChoices = new String[10]; //Maximum 10 global questions
             allowed.[0-9]
20
21    }
22    public String run(String quiz) {
23       qsource = quiz;
24       errorMsg = "";
25       qsource += '\n'; //Sometimes it is necessary to look ahead. Adding an
             extra char to avoid array out of bound.
26       System.out.println("Parser received:");
27       System.out.print(qsource);
28       cursor = 0;
29       marker = 0;
30       error = false;
31
32       while(cursor < qsource.length()) {
33          switch(qsource.charAt(cursor) ) {
34          case '<':
35             skip_html_tag();
36             break;
37          case '\\': //Note: that is just one \, the other one is for java
                compiler to escape it.
38             escape();
39             break;
40
41          case '=':
42             fill_in_blank();
43             if(error)
44                return errorMsg;
45             break;
46
47          case '*':
```

```
48            hint();
49            if(error)
50               return errorMsg;
51            break;
52
53         case '(':
54            pop_up_hint();
55            if(cursor < qsource.length() && qsource.charAt(cursor) == '|') {
56               parseError("Got to a '|' without seeing a '('");
57               cursor++;
58               break;
59            }
60            if (cursor < qsource.length() && qsource.charAt(cursor) == ')' )
                  {
61               parseError("Got to a ') without the rest of expression.");
62               cursor++;
63               break;
64            }
65            else
66               cursor++;
67            if(error)
68               return errorMsg;
69            break;
70            //case ')':
71            //case '|':
72            //parseError("Incomplete \"pop up hint\" expression. Must be in
                 format (word|hint)");
73            //i++;
74            //break;
75         case '[':
76            inline_mult_question();
77            if(error)
78               return errorMsg;
79            break;
80         case '.':
81            if ((qsource.length() > 2) && (qsource.charAt(0) == '.' &&
                 qsource.charAt(1) == 'c')) //the line must start with .c
```

```
82              global_mult_choice_options();
83
84            if((cursor+1 < qsource.length()) && (qsource.charAt(cursor) == '
                  .' && qsource.charAt(cursor+1) == 'c')) {
85              parseError("Global mutliple choices must be on a line by
                    themselves.<br>" +
86                  "ie: The line starts with \".c\" and ends with the last
                        choice. For example:<br>" +
87                  ".c choice1 choice2 ... last_choice");
88              cursor++; //Read '.'
89              return errorMsg;
90            }
91            cursor++; //Read '.'
92            if(error)
93              return errorMsg;
94            break;
95          case '#':
96            global_mutl_choice_question();
97            if(error)
98              return errorMsg;
99            break;
100         default:
101           cursor++;
102         }//switch
103       }//while
104       return "";//no error
105
106   }//run
107
108   /*********************************
109    * Skipping over the possible html tags
110    *********************************/
111
112   /*
113    * Escape over the html tags ie: <html> is ignored by the parser
114    * Since we are not actually writing a html parser, this method
115    * only works with simple html tags.*/
```

```
116    private void skip_html_tag() {
117       cursor++; //read '<'
118       int start = cursor;
119
120       char c;
121       int len = qsource.length();
122       for(; cursor < len; cursor++) {
123          c = qsource.charAt(cursor);
124          if (c == '>'){
125             cursor++; //read '>'
126             showHint(start, cursor);
127             return; //we got to the end of the html tag
128          }
129
130       }
131
132       /* If we get here, it means we scanned the entire file and
133        * did not see a '>' to close the html tag, so we assume it
134        * was not an html tag to start with. ie, 5 < 7
135        * Reset the counter and continue with parsing
136        * */
137       System.out.println("No html tag here");
138       cursor = start;
139    }
140
141    /***********************************
142     * Escape character. ie. \=word will
143     * not invoke the fill in the blank.
144     ***********************************/
145    /*
146     * option 1: (implemented) increment the read counter pass the trigger
            keyword.
147     * option 2: (not implemented) do follow the trigger keyword,
148     * and when it returns either ignore the errors or the actual question
            .*/
149    private void escape() {
150
```

```
151       cursor++; //read the escape char, '\'
152       cursor++; //read the character after the '\'
153
154    }
155
156    /*********************************
157     * Hint questions
158     *********************************/
159    private void hint() {
160       int start = cursor;
161       cursor++; //read *
162       int len= qsource.length();
163       char c;
164
165       for( ; cursor < len ; cursor++) {
166          c =  qsource.charAt(cursor);
167          if( c == '*') { //Could've put it inside a for statement, but
                 easier to read this way
168             showHint(start,cursor);
169             cursor++; //read the closing '*', otherwise the function will
                    get called again
170             return;
171          }
172          if( Character.isWhitespace(c) ) {
173             parseError("Hint questions: Spaces are not allowed.<br>");
174             return;
175          }
176
177       }
178
179       parseError("Reached the end of file but did not see another '*' to
              close the \"Hint\" question.<br>");
180       return;
181    }
182
183    /*********************************
184     * Global multiple-choice questions
```

```
185     * ******************************/
186    /*
187     * regexp:  "([0-9])#([^ .,;\n])([ .,;\n]))"
188     *         "#([^ .,;\n])([ .,;\n]))"
189     * <global_multiple_choices_question> := <digit>? "#" not<terminator> <
            answer> <terminator>
190     * <digit> := [0-9]
191     * <answer> := any characters except the terminator chars
192     * <terminator> := ' '|.|,|;|\n
193     * */
194    //#choice
195    //1#choice
196    private void global_mutl_choice_question() {
197       int start = cursor;
198       int startOfAnswer;
199       int digit = 0;
200
201       if((cursor-1 >= 0)) { //check to see if '#' sign was followed by a
              digit.
202          digit = getDigitAt(cursor-1);
203          if(digit == -1)
204             digit = 0; //fall back on the default value. There was no digit
                   before '#' sign
205       }
206
207       cursor++; //read the '#'
208       startOfAnswer = cursor;
209
210       if(global_choice_quesion_terminator() == true)
211       {
212          parseError("Global multi choices question<br>\"#\" can not be
                followed by any of '.' or ',' or ';' or 'new line' or space<br
                >");
213          return;
214       }
215       answer();
216       if(global_choice_quesion_terminator() == false){
```

```
217         parseError("Global Multiple Choice question<br>Reached the end of
                input but the word did not terminate.");
218         return;
219       }
220       String answer = qsource.substring(startOfAnswer, cursor);
221       if(answer == null){
222         parseError("Global Mulitple Choice Question<br>There must be an
                answer after '#' sign.");
223         return;
224       }
225       checkInTable(digit, answer);
226       showHint(start, cursor);
227
228
229     }
230
231     private void answer(){
232
233       for(; cursor< qsource.length(); cursor++){
234         if(global_choice_quesion_terminator() == true)
235           return;
236       }
237       parseError("Global Multiple Choice quesion<br>Reached the end of input
              but the word did not terminate.");
238     }
239
240     private boolean global_choice_quesion_terminator() {
241       switch (qsource.charAt(cursor)) {
242       case ' ':
243         return true;
244       case '\n':
245         return true;
246       case '\r':
247         return true;
248       case '.':
249         return true;
250       case ',':
```

```
251         return true;
252       case ';':
253         return true;
254
255       }
256       return false; // did not have any of the terminating chars
257
258     }
259     /*
260      * Guaranteed: answer is not null*/
261     private void checkInTable (int num, String answer) {
262       if(globalMultChoices[num] == null)
263       {  //                          .c# or .c is replaced here
264         parseError("There aren't any '.c"+(num==0?"":num)+"' that include \
              ""+answer+"\"");
265         return;
266       }
267
268       //check each against the answer.
269       Pattern pattern = Pattern.compile("\\b"+answer+"\\b");
270       Matcher matcher = pattern.matcher(globalMultChoices[num]);
271       boolean isAmongChoices = matcher.find();
272       if(! isAmongChoices) {
273         parseError("The answer \""+answer+"\" is not amongst any of the
              choices given by '.c"+(num==0?"":num)+"'" +
274               "<br>The choices are: "+globalMultChoices[num]);
275         return;
276       }
277       //no error, return.
278     }
279
280
281     /*********************************
282      * Global multipl-choice options
283      * ********************************/
284     /*.c choice1 choice2 ... last_choice
285      *.c# choice1 choice2 ... last_choice
```

```
286     **/
287     private void global_mult_choice_options() {
288          numChoices = 0;
289          cursor++; //read the '.'
290          cursor++; //read 'c'
291
292          int digit = digit();
293          if(digit<0 || digit>10){
294             parseError("The numbers in global multiple choice quesions
                    should be in the range of 0 to 9.<br>You typed: "+digit);
295             return;
296          }
297          int start = cursor;
298          if(qsource.charAt(cursor) != ' '){
299             parseError("Global Multiple Choices<br>Need a space after .c or
                    .c# <br>ie: .c choice1 choice2 ... last_choice\n");
300             return;
301          }
302          cursor++; //read the space char.
303          choices();
304          if(numChoices < 1){
305             parseError("Global Multiple Choice<br>There must be at least one
                     choice.\n");
306             return;
307          }
308
309          addChoicesToTable(digit, qsource.substring(start, cursor));
310          if(error)
311             return;
312
313          showHint(start, cursor);
314          System.out.println("choices are: "+qsource.substring(start,cursor)+
                " at index: "+digit);
315     }
316     private void choices() {
317          //The entire process of going through choices can be done with the
                following
```

```
318         //while loop. But I opted out of it so the code more closely
                resemble the
319         //BNF notation.
320        /*while(i < input.length()) {
321            if(input.charAt(i) == '\n')
322               return;
323         }*/
324         char c;
325         int tmp = 0;
326         for(; cursor< qsource.length(); cursor++){
327            tmp = glob_mult_choice_option_word();
328            if(tmp > 0) {
329               numChoices++;
330            }
331
332            c = qsource.charAt(cursor);
333            if(c == '\n') {//end of choices
334               //add the choices to the table
335               return;
336            }
337            else if(c == ' ') { //have more choices
338               continue;
339            }
340            else{ //error
341               parseError("Global Multiple Choice<br>Choices must be space
                      separated and new line terminated.<br>");
342               return;
343            }
344         }
345         //end of file or input
346         parseError("Reached end of input but did not terminate the Global
                Multiple Choices.");
347         return;
348       }
349
350       /*
351        * <word> := "more than one character length" "space separated"
```

```
352         */
353        private int glob_mult_choice_option_word() {
354           int countChars = 0; //counts the number of characters in each word
355           char c = qsource.charAt(cursor);
356
357           for(; cursor < qsource.length(); cursor++){
358              c = qsource.charAt(cursor);
359              if(Character.isWhitespace(c)){ //reached the end of a word
360                 return countChars;
361              }
362              countChars++;
363           }
364
365           return countChars;
366        }
367
368        /*
369         * <digit> := [0-9]
370         */
371        private int digit() {
372           int digit = 0; //The default value
373
374           //read the digit
375           if(Character.isDigit(qsource.charAt(cursor))) {
376              int c = Character.codePointAt(qsource, cursor);
377              digit = Character.digit(c, 10);
378              cursor++; //read the digit
379              return digit;
380           }
381           //There was no digit to read. Return the default value
382           return 0;
383        }
384
385        private int getDigitAt(int pos) {
386           int digit = -1; //-1 is error if returned.
387           if(Character.isDigit(qsource.charAt(pos))) {
388              int c = Character.codePointAt(qsource, pos);
```

```
389            digit = Character.digit(c, 10);
390            return digit;
391         }
392         return -1; //return error, make sure the calling method checks it.
393      }
394      private void addChoicesToTable(int num, String choices) {
395         if(globalMultChoices[num] == null) {
396            globalMultChoices[num] = choices;
397
398         }else { //error
399            if(num == 0) {
400               parseError("There's already a \".c\" in use. Consider adding
                      a number after 'c'");
401            }else {
402               parseError("The number "+num+" in \".c"+ num +"\" is already
                      used by another multiple choice.<br>Consider changing the
                      number.");
403            }
404
405         }
406
407      }
408
409
410   /**********************************
411    * inline multiple choice question
412    **********************************/
413   /*
414    * <question> := '[' <word>* @<word> <word>* ']'
415    * <word> := "any chars except @" "space"
416    */
417   private void inline_mult_question() {
418      //Received the opening '[', on to the next part
419      int start = cursor, end = 0;
420
421      cursor++; //read the '['
422      while(word()) cursor++;
```

```
423
424       if(cursor >= qsource.length() || qsource.charAt(cursor) != '@'){
425          parseError("Expected @ to indicate the answer for the in-line
                multiple choice question");
426          return;
427       }
428
429       cursor++; //read @
430       word();
431       while(word()) cursor++;
432       if(cursor >= qsource.length() || qsource.charAt(cursor) == '@') {
433          parseError("Only 1 answer is allowed among the choices.");
434          return;
435       }
436       if(cursor >= qsource.length() || qsource.charAt(cursor) != ']' ){
437          parseError("Expected ] to terminate the in-line multiple-choice
                question");
438          return;
439       }
440       cursor++; //read ]
441       end = cursor;
442       showHint(start,end);
443    }
444    /*
445     * <word> := "any character expect @ or ]" "space"
446     */
447    private boolean word(){
448       char c;
449       while(cursor < qsource.length() ) {
450          c = qsource.charAt(cursor);
451          if (Character.isWhitespace(c))
452             return true;;
453          if(c == '@'){
454             return false;
455          }
456          if (c == ']') {
457             return false;
```

```
458         }
459         cursor++; //on to the next character
460       }
461       parseError("Got to the end of the input without terminating the in-
              line mulitple-choice question.");
462       return false;
463
464     }
465
466
467     /*********************************
468      * pop up hint
469      *********************************/
470     /*<pop_up_hint> := "(" <spaces>* <word> <spaces>* "|" <spaces>* <word> <
           spaces>* ")"
471      * <word> := anything.
472      *
473      * ( word | word )*/
474     private void pop_up_hint() {
475       int start = 0, end = 0;
476       start = cursor;
477       errInWord = false;
478       cursor++; //read the (
479       while(qsource.charAt(cursor) == ' ' ) cursor++; //ignore the white
              spaces
480
481       pop_up_hint_word();
482       if(errInWord) return;
483
484       while(qsource.charAt(cursor) == ' ' ) cursor++; //ignore the white
              spaces
485
486       if(qsource.charAt(cursor) == '|')  //got to the separating '|'
487         cursor++;
488       else{//error
489         parseError("Only one word is allowed on either side of \"|\"<br>");
490         parseError("Pop_up_hints are \"|\" seperated.");
```

```
491         return;
492       }
493
494       while(qsource.charAt(cursor) == ' ' ) cursor++; //ignore the white
              spaces
495
496       pop_up_hint_word();
497       if(errInWord) return;
498
499       while(qsource.charAt(cursor) == ' ' ) cursor++; //ignore the white
              spaces
500
501       if(qsource.charAt(cursor) == ')')  //got to the closing ')'
502         cursor++;
503       else{//error
504         parseError("Only one word is allowed on either side of \"|\"<br>");
505         parseError("Pop_up_hints are \")\" terminated.");
506         return;
507       }
508
509       end = cursor;
510       showHint(start, end);
511     }
512
513     private void pop_up_hint_word() {
514       int start = cursor;
515       if(errInWord)
516         return;
517
518       for (; cursor < qsource.length(); cursor++) { //keep reading until a
              "|" or ")"
519         char c = qsource.charAt(cursor);
520         if (c == ' ') {
521           return;
522         }
523         if (c == '('){
524           parseError("Can not have inner expressions.");
```

```
525             errInWord = true;
526             return;
527          }
528          if (c == '|' || c == ')'){
529             if (cursor == start){
530                parseError("The 'Hint' words can not be empty.");
531                errInWord = true;
532                return;
533             }
534             return; //no errors;
535          }
536
537       }
538
539       parseError("Error: Pop_up_hints.<br>Got to the end of line and didn't
              see a \"|\" or \")\" <br>");
540       errInWord = true;
541       return;
542
543    }
544
545    /*************************
546     * fill-in-the-blanks
547     *************************/
548    private void fill_in_blank() {
549       int start = 0, end = 0;
550       start = cursor;
551       cursor++; // read the = sign
552       int numChar = fill_in_blank_word();
553       if (numChar == 0) {
554          parseError("Empty fill in blanks not allowed.<br>ie, '=' can not be
                 followed by space" +
555                   " or '.' or ',' or '=' or new line.");
556          return;
557       }
558       terminator();
559       end = cursor;
```

```
560
561       showHint(start, end);
562
563
564    }
565
566    private int fill_in_blank_word() {
567       int count = 0;
568       for (; cursor < qsource.length(); cursor++) { //keep reading until a
              terminator char
569          char c = qsource.charAt(cursor);
570          if (c == ';') {// error
571             parseError("fill-in-blanks can not contain ';'");
572             return 0;
573             //it returns 0 which signals there's no char after = and prints
                  another error
574             //message.
575          }
576          if (c == ' ' || c == '.' || c == ',' || c == '\n' || c == '=') { //
                the terminator characters.
577             return count;
578          }
579          count++;
580       }
581       return count;
582    }
583
584    private void terminator() {
585       char c = qsource.charAt(cursor);
586       if (c == ' ' || c == '.' || c == ',' || c == '\n') { // the terminator
             characters. sanity check
587          cursor++; // read
588          return;
589
590       } else if(c == '='){
591          parseError("Fill-in-the-blanks must be space separated.<br>ie: =
                word =word");
```

```
592         return;
593       }
594       else // error
595         parseError("Fill-in-the-blanks must end with a space, or '.' or ','
                or a new line.");
596     }
597
598     private void showHint(int start, int end) {
599
600       for (int j = marker; j < start; j++)
601         System.out.print(qsource.charAt(j));
602       for (int j = start; j < end; j++) //it'll put one for the deliminator
            char too.
603         System.out.print("*");
604
605       marker = end;
606     }
607
608     private void skipSpaces() {
609       while(qsource.charAt(cursor) == ' ') {
610         cursor++;
611       }
612     }
613     private void parseError(String errMsg) {
614       /* Change "error" to true for production.
615        * If false, it will continue parsing even if there's an error.*/
616       errorMsg = "ERROR: "+ errMsg;
617       error = true;
618     }
619 }
```

A.2 Parser MMP Injector

```
1  package modelLayer.parser;
2
3  /**
4   * This is the second pass of the quiz input.
```

```
 * it will look through the input and insert mmp instead of the input
 */
public class QuizScriptMMPInjector {

   StringBuffer qsource; //the input from user
   int cursor;

   public QuizScriptMMPInjector (String quiz) {
      qsource = new StringBuffer(quiz);
      qsource.append('\n');
      cursor = 0;
   }
   public QuizScriptMMPInjector () {
      cursor = 0;
   }

   public StringBuffer run (String quiz) {
      qsource = new StringBuffer();
      qsource.append(quiz);
      qsource.append('\n');
      cursor = 0;
      return run();
   }
   public StringBuffer run() {
      System.out.println("MMP Injector recieved: "+qsource);

      while(cursor < qsource.length()) {
         switch(qsource.charAt(cursor) ) {
         case '<':
            skip_html_tag();
            break;
         case '\\':
            escape();
            break;

         case '=':
            fill_in_blank();
```

```
42            break;
43          case '*':
44            hint();
45            break;
46          case '(':
47            pop_up_hint();
48            break;
49  //      case ')':
50  //      case '|':
51  //        parseError("Incomplete \"pop up hint\" expression. Must be in
        format (word|hint)");
52  //        i++;
53  //        break;
54          case '[':
55            inline_mult_question();
56            break;
57
58            //TODO: the choices must be at the beginning of the line and one
                   line at a time.
59          case '.':
60            if ((qsource.length() > 2) && qsource.charAt(cursor+1) == 'c')
                  //the line must start with .c
61              global_mult_choice_options();
62            else{
63              cursor++;
64            }
65            break;
66          case '#':
67            global_mutl_choice_question();
68            break;
69          case 'q': //qq case
70            if ((qsource.length() > 2) && qsource.charAt(cursor+1) == 'q')
71              automatic_numbering();
72            else{
73              cursor++;
74            }
75            break;
```

```
76
77
78          default:
79             cursor++;
80          }//switch
81       }//while
82       return qsource;
83    }//run
84
85    /* Hidden feature of quizscript: automatic numbering
86     * create mmp : {qq/}
87     * */
88    private void automatic_numbering() {
89       int start = cursor;
90       cursor++; //read the first q
91       qsource.replace(start, cursor+1, "{qq/}");
92       System.out.println(qsource);
93       cursor++; //read the second q
94
95    }
96    /*
97     * Skip over the html tags ie: <html> is ignored by the parser
98     * Since we are not actually writing an html parser, this method
99     * only works with simple html tags.*/
100   private void skip_html_tag() {
101      cursor++; //read '<'
102      int start = cursor;
103
104      char c;
105      int len = qsource.length();
106      for(; cursor<len; cursor++) {
107         c = qsource.charAt(cursor);
108         if (c == '>'){
109            cursor++; //read '>'
110            return; //we got to the end of the html tag
111         }
112
```

```
113       }
114
115       /* If we get here, it means we scanned the entire file and
116        * did not see a '<' to close the html tag, so we assume it
117        * was not an html tag to start with. ie, 5 < 7
118        * Reset the counter and continue with parsing
119        * */
120       cursor = start;
121    }
122
123    //skip the trigger symbol.
124    private void escape() {
125       qsource.deleteCharAt(cursor);
126       cursor++; //read the char after it
127
128    }
129
130    /*
131     * read from * to *, extract the word (anything in the middle)
132     * create mmp: "{b1 word/}"
133     */
134    private void hint() {
135       int start = cursor;
136       cursor++; //read *
137       char c;
138       int len = qsource.length();
139
140       do{
141          c = qsource.charAt(cursor);
142          cursor++;
143
144       }while(cursor < len && c != '*');
145       CharSequence word = extract(start+1, cursor-1);
146
147       qsource.replace(start, cursor, "{bl "+word+"/}");
148
149       System.out.println(qsource);
```

```
150    }
151    private void global_mutl_choice_question() {
152       int start = cursor;
153       CharSequence digit = "";
154
155       if( cursor-1 >= 0 && Character.isDigit(qsource.codePointAt(cursor-1)))
              {
156          digit = extract(start-1,cursor);
157          System.out.println("digit :"+digit);
158          start = cursor-1;
159       }
160
161       cursor++; //read the '#'
162
163       char c;
164       int len = qsource.length();
165       int startOfWord = cursor;
166       do{
167          c = qsource.charAt(cursor);
168          cursor++;
169       }while(cursor < len && (c != ' ') && (c != '.') && (c != ',') && (c !=
              '\n') && (c != '\r') && (c != ';'));
170
171       CharSequence word = extract(startOfWord, cursor-1);
172       System.out.println("word :"+ word);
173
174       qsource.replace(start, cursor, "{pq_ "+word+" "+digit+ "/} "); //+" "+
              digit+"/}\"");
175
176       System.out.println(qsource);
177    }
178
179    /*
180     * read from ".c#" to end of line, extract the_choices ( everything in
            the middle).
181     * The digit that follows .c is optional.
182     * create mmp: if there's a digit: "{choicesi # the_choices/}"
```

```
183     * if there's no digit: "{choices the_choices/}"
184     * */
185    private void global_mult_choice_options() {
186       int start = cursor;
187       cursor++; //read .
188       cursor++; //read c
189
190       boolean hasDigits = false;
191       if(Character.isDigit(qsource.codePointAt(cursor))) {
192          cursor++; //read the digit
193          hasDigits = true;
194       }
195
196       char c;
197       int len = qsource.length();
198       do{
199          c = qsource.charAt(cursor);
200          cursor++;
201       }while(cursor < len && (c != '\r') && (c != '\n'));
202
203       CharSequence words = extract(start+2, cursor-1);
204
205       //insert mmp
206       if(hasDigits) {
207          qsource.replace(start, cursor, "{choicesi "+words+"/}"); //it also
               includes the digit
208       }else{
209          qsource.replace(start, cursor, "{choices "+words+"/}");
210       }
211       System.out.println(qsource);
212
213    }
214
215    /*
216     * read from [ to ], extract word (anything in the middle)
217     * create mmp: "{cq word/}"
218     */
```

```
219    private void inline_mult_question() {
220       int start = cursor;
221       cursor++; //read [
222       char c;
223       int len = qsource.length();
224
225       do{
226          c = qsource.charAt(cursor);
227          cursor++;
228
229       }while(cursor < len && c != ']');
230       CharSequence words = extract(start+1, cursor-1);
231
232       qsource.replace(start, cursor, "{cq "+words+"/}");
233
234       System.out.println(qsource);
235    }
236
237    /* read from ( to |, extract the word
238     * read from | to ), extract the hint
239     * create mmp : "{tt word hint}"
240     * insert mmp into quiz
241     * */
242    private void pop_up_hint() {
243       int start = cursor;
244       cursor++; //read '('
245       char c;
246       int len = qsource.length();
247       CharSequence word = "", hint = "";
248
249       do{
250          c = qsource.charAt(cursor);
251          cursor++;
252       }while(cursor < len && c != '|');
253
254       word = extract(start+1, cursor-1);
255       System.out.println("word :"+word);
```

```
256
257       int startOfHint = cursor;
258
259       do{
260          c = qsource.charAt(cursor);
261          cursor++;
262       }while(cursor < len && c != ')');
263
264       hint = extract (startOfHint, cursor-1);
265       System.out.println("hint :"+hint);
266
267       //Create mmp "{tt $1 $2/}"
268       String mmp = "{tt $1 $2/}";
269       mmp = mmp.replace("$1", word);
270       mmp = mmp.replace("$2", hint);
271
272       //Insert mmp
273       qsource.replace(start, cursor, mmp);
274       System.out.println(qsource);
275    }
276
277    //keep reading until you hit a terminating char
278    private void fill_in_blank() {
279       cursor++; //read '='
280       int start = cursor;
281       char c;
282
283       do{
284          c = qsource.charAt(cursor);
285          cursor++;
286       }while(cursor<qsource.length() && (c != ' ') && (c != '.') && (c !=
             ',') && (c != '\n') && (c != '\r') && (c != '=') && (c != ';'));
287
288       CharSequence word = extract (start, cursor-1); //extract word
289
290       //String mmp = " \"{tr $1/}\"";
291       //mmp = mmp.replace("$1", word); //results in: " {tr someWord}"
```

```
292       //qsource.replace(start-1, cursor-1, mmp);
293       qsource.replace(start-1, cursor-1, " {tr "+word+"/}"); //Equivalent to
              the top 3 lines.
294
295       System.out.println(qsource);
296    }
297
298    private CharSequence extract (int start, int end) {
299       return (qsource.subSequence(start, end));
300    }
301 }
```

A.3 Database Communication Classes

```
1  package modelLayer.db;
2
3  /*
4   * Database connectivity.
5   * creates the required connection, statement, runs the query, and gets the
        result from db. Then closes the resources.
6   *
7   * */
8  /* thanks to: http:// tomcat.apache.org/ tomcat-6.0-doc/ jndi-datasource-
       examples-howto.html
9   * thanks to: http:// www.ntu.edu.sg/ home/ ehchua/ programming/ java/
        JavaWebDBApp.html*/
10
11 import java.io.IOException;
12 import java.sql.Connection;
13 import java.sql.ResultSet;
14 import java.sql.SQLException;
15 import java.sql.PreparedStatement;
16 import java.util.Vector;
17
18 import javax.naming.InitialContext;
19 import javax.naming.NamingException;
20 import javax.servlet.ServletConfig;
```

```
21 import javax.servlet.ServletException;
22 import javax.servlet.http.HttpServlet;
23 import javax.servlet.http.HttpServletRequest;
24 import javax.servlet.http.HttpServletResponse;
25 import javax.sql.DataSource;
26
27 //import com.mysql.jdbc.PreparedStatement; this will cause problem, do not
       use
28
29 public class GetData extends HttpServlet {
30    private static final long serialVersionUID = 1L;
31
32    private DataSource pool; // pool of db connections
33
34     public GetData() {
35         super();
36     }
37
38    public void init(ServletConfig config) throws ServletException {
39       System.out.print("GetData->init");
40       super.init(config); // calling the super.init so the class and its
             parent will be able to access the ServletConfig during the
             lifetime to the servlet.
41
42       try { //get DataSource
43          InitialContext initContext = new InitialContext(); //Create a JNDI
                Initial context to be able to lookup the DataSource
44          //Lookup the DataSource for the database
45          pool = (DataSource) initContext.lookup( "java:comp/env/jdbc/
                quizscript" );
46          if (pool == null) {
47             throw new ServletException("Can't find 'jdbc/quizscript' in the
                   pool.");
48          }
49       } catch (NamingException e) {
50          System.out.println(e);
51          log(""+e);
```

```
52      }
53    }
54
55    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
56      System.out.print("in GetData ...");
57
58      boolean errFlag = false;
59      //create vectors to hold retrieved data
60      //'beans' would've been a good idea too.
61      Vector<String> id_vec = new Vector<String>();
62      Vector<String> desc_vec = new Vector<String>();
63      Vector<String> views_vec = new Vector<String>();
64      Vector<String> dates_vec = new Vector<String>();
65      Vector<String> sources_vec = new Vector<String>(); //debugging
            purposes
66
67      Connection con = null;
68      PreparedStatement prepStatement = null;
69      ResultSet results = null;
70
71      String howMany = (String) request.getAttribute("howMany");
72      String offset = (String) request.getAttribute("offset");
73      String quizID = (String) request.getAttribute("quizID");
74      Boolean updateReqd = (Boolean) request.getAttribute("update");
75      if(updateReqd == null) updateReqd = false;
76
77      String query;
78      if (quizID != null) {
79        query = "SELECT * FROM testTable WHERE id = "+quizID;
80      }else {
81        query = "SELECT * FROM testTable ORDER BY id DESC LIMIT " + offset
              + " , " + howMany;
82      }
83      if(updateReqd && quizID != null) {
84        query = "UPDATE testTable SET viewCount = viewCount +1 WHERE id =
              "+quizID;
```

```
85      }
86      System.out.println(query);
87      try {
88         con = pool.getConnection();
89         prepStatement = con.prepareStatement(query);
90         if(updateReqd){
91            int updateResult = prepStatement.executeUpdate();
92            System.out.println("updated "+updateResult+" rows.");
93         }else{
94            results = prepStatement.executeQuery();
95         }
96
97         if (updateReqd == false) { //continue only if we wanted to access
               the database for its data
98
99            if (results.next()) { //read the data
100              do { //get data
101                 id_vec.add( results.getString("id") );
102                 desc_vec.add ( results.getString("description") );
103                 views_vec.add ( results.getString("viewCount") );
104                 dates_vec.add ( results.getString("dateCreated") );
105                 sources_vec.add ( results.getString("source") );
106                 //System.out.println( "GetData: source: " + sources_vec.
                       lastElement() );
107              }while(results.next());
108           }else { //no data
109              setError(request, errFlag, "No results returned from database
                     .");
110           }
111        }
112
113     }catch (SQLException e) {
114        System.out.println("SQL error "+e);
115        log("SQL error "+e);
116        setError(request, errFlag, e.toString());
117     }catch (Exception e) {
118        System.out.println("General error "+e);
```

```
119         log("General error "+e);
120         setError(request, errFlag, e.toString());
121       }finally { //closing time.
122         if (results != null) { try { results.close(); } catch(SQLException
              e) { log ("ResultSet did not close."+e); setError(request,
              errFlag, e.toString()); }}
123         if (prepStatement != null) { try {prepStatement.close();} catch(
              SQLException e) {log("Statement did not close."+e); setError(
              request, errFlag, e.toString()); }}
124         if (con != null) {try { con.close(); } catch(SQLException e) {log("
              db connection did not close"+e); setError(request, errFlag, e.
              toString()); }}
125         //connection is not actually closed, rather is send back to the
              pool for reuse.
126
127       }
128
129       if(errFlag)
130       {
131         System.out.println("*** SQL ERROR. ***");
132         return;
133       }
134
135       //now let's give the data back to the calling servlet
136       if(updateReqd == false) {
137         request.setAttribute("id_vec", id_vec);
138         request.setAttribute("desc_vec", desc_vec);
139         request.setAttribute("views_vec", views_vec);
140         request.setAttribute("dates_vec", dates_vec);
141         request.setAttribute("sources_vec", sources_vec);
142       }
143
144     }
145
146     /* Sets the error message only for the first exception/error that
          happens.
147      * the subsequent ones are ignored.*/
```

```
148    private void setError(HttpServletRequest request, boolean errFlag,
           String errorMsg) {
149       if(errFlag == false) {
150          errFlag = true;
151          request.setAttribute("ErrorMsg", errorMsg);
152       }
153    }
154
155    protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
156       doPost(request, response);
157    }
158
159 }
```

```
 1 package modelLayer.db;
 2
 3 import java.io.IOException;
 4 import java.sql.Connection;
 5 import java.sql.ResultSet;
 6 import java.sql.SQLException;
 7 import java.sql.PreparedStatement;
 8 import java.sql.Statement;
 9 import java.util.Date;
10
11 import javax.naming.InitialContext;
12 import javax.naming.NamingException;
13 import javax.servlet.ServletConfig;
14 import javax.servlet.ServletException;
15 import javax.servlet.http.HttpServlet;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18 import javax.sql.DataSource;
19
20 public class SaveData extends HttpServlet {
21    private static final long serialVersionUID = 1L;
22
```

```
23    private DataSource pool; // pool of db connections
24
25     public SaveData() {
26         super();
27     }
28
29    public void init(ServletConfig config) throws ServletException {
30       System.out.print("SaveData->init");
31       super.init(config); // calling the super.init so the class and its
             parent will be able to access the ServletConfig during the
             lifetime to the servlet.
32
33       try { //get DataSource
34          InitialContext initContext = new InitialContext(); //Create a JNDI
                Initial context to be able to lookup the DataSource
35          //Lookup the DataSource for the database
36          pool = (DataSource) initContext.lookup("java:comp/env/jdbc/
                quizscript");
37          if (pool == null) {
38             throw new ServletException("Can't find 'jdbc/quizscript' in the
                   pool.");
39          }
40       } catch (NamingException e) {
41          System.out.println(e);
42          log(""+e);
43       }
44    }
45
46    protected void doPost(HttpServletRequest request, HttpServletResponse
         response) throws ServletException, IOException {
47       System.out.println("SaveData...");
48
49       Connection con = null;
50       PreparedStatement prepStatement = null;
51       ResultSet results = null;
52
53       String errorMsg = "";
```

```
54
55      String authorName = (String) request.getAttribute("authorName");
56      String quizSrc = (String) request.getAttribute("quizSrc");
57      String quizDesc = (String) request.getAttribute("quizDesc");
58
59      String query =  "INSERT INTO testTable ( description, viewCount,
            dateCreated, source, authorName ) VALUES(?,?,?,?,?)";
60
61      try {
62         con = pool.getConnection();
63         prepStatement = con.prepareStatement(query, Statement.
               RETURN_GENERATED_KEYS);
64         prepStatement.setString(1, quizDesc); //description
65         prepStatement.setInt(2, 0); //viewCount
66         //Get the date to be inserted in db
67         Date nowDate = new Date();
68         long date = nowDate.getTime();
69         java.sql.Date sqlDate = new java.sql.Date(date);
70         prepStatement.setDate(3, sqlDate);
71
72         prepStatement.setString(4, quizSrc); //Quiz source
73         prepStatement.setString(5, authorName); //author's name
74         System.out.println(prepStatement.toString());
75
76         prepStatement.executeUpdate();
77         //get the id for the newly created quiz.
78         //since running on the same session, dbms will make sure there not
               any concurrency.
79         results = prepStatement.getGeneratedKeys();
80         int createdQuizID = -1;
81         if(results != null && results.next()){
82            createdQuizID = results.getInt(1);
83            request.setAttribute("createdQuizID", createdQuizID);
84            request.setAttribute("errorMsg", errorMsg);
85            System.out.println("Created quiz ID:"+createdQuizID);
86         }
87
```

```
88        }catch (SQLException e) {
89           request.setAttribute("errorMsg", errorMsg);
90           System.out.println("SQL error "+e);
91           log("SQL error "+e);
92        }catch (Exception e) {
93           request.setAttribute("errorMsg", errorMsg);
94           System.out.println("General error "+e);
95           log("General error "+e);
96        }finally { //closing time.
97           if (results != null) { try { results.close(); } catch(SQLException
                e) { log ("ResultSet did not close."+e); }}
98           if (prepStatement != null) { try {prepStatement.close();} catch(
                SQLException e) {log("Statement did not close."+e); }}
99           if (con != null) {try { con.close(); } catch(SQLException e) {log("
                db connection did not close"+e);}}
100          //connection is not actually closed, rather is send back to the
                pool for reuse.
101       }
102    }
103
104    protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
105       doPost(request, response);
106    }
107 }
```

Appendix B

Implementation of Controller Layer

The following sections include the source code for the Java classes that make up the Controller Layer.

B.1 Home Page Servlet

```
1  package controllerLayer;
2
3  /**
4   * Get the quizzes from database and pass it to the view layer
5   */
6  import java.io.IOException;
7
8  import javax.servlet.RequestDispatcher;
9  import javax.servlet.ServletContext;
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15 public class HomePageServlet extends HttpServlet {
16    private static final long serialVersionUID = 1L;
17    //parameters used for 'limit' clause of MySQL
18    //eg. SELECT * FROM some_table LIMIT 5, 10; # retrieves rows 6 to 15
```

```
19    private static final String OFFSET_VAL = "0"; //'offset' parameter, ie,
          which row to start from
20    private static final String HOWMANY_VAL = "5"; //how many rows to return
21
22     public HomePageServlet() {
23         super();
24     }
25
26     protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
27       System.out.println("in HomePageServlet...");
28
29       //from which row and how many quizzes is the user asking for.
30       String offset = request.getParameter("offset");
31       String howMany = request.getParameter("howMany");
32
33       //Sanity check. Use default values
34       if(offset == null) {
35          offset = OFFSET_VAL;
36       }else {
37          offset = (Integer.parseInt(offset)+5)+"";
38          //offset = OFFSET_VAL;
39       }
40       if(howMany == null) {
41          howMany = HOWMANY_VAL;
42       }
43
44       //adding more info to the request and calling the model layer
45       request.setAttribute("offset", offset);
46       request.setAttribute("howMany", howMany);
47
48       System.out.println("redirecting to 'GetData'");
49       ServletContext context = getServletContext();
50
51       RequestDispatcher dispatcher = context.getNamedDispatcher("GetData");
52       dispatcher.include(request, response); //go to model layer and come
             back
```

```
53
54        String errorMsg = (String) request.getAttribute("ErrorMsg");
55        if(errorMsg == null || errorMsg.equals("")) {
56           passToDiffModule(request, response, "HomePage");
57        }else {
58           passToDiffModule(request, response, "Error");
59        }
60
61     }
62     private void passToDiffModule(HttpServletRequest request,
           HttpServletResponse response, String servletName ) throws
           ServletException, IOException {
63        RequestDispatcher dispatcher;
64        ServletContext context = getServletContext();
65        dispatcher = context.getNamedDispatcher(servletName);
66
67        if(dispatcher == null) {
68           log("Quiz module couldn't find the requested servlet: "+servletName
              );
69
70           request.setAttribute("ErrorMsg", "Could not find the module: "+
              servletName);
71           dispatcher = context.getNamedDispatcher("Error");
72           if(dispatcher == null){
73              System.out.println("Can not find the 'Error.jsp' module."); //
                 Nothing else can be done.
74              return;
75           }
76        }
77
78        dispatcher.forward(request, response);
79        return;
80
81     }
82
83     protected void doGet(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
```

```
84       doPost(request, response);
85    }
86
87 }
```

B.2 Parser Servlet

```
1  package controllerLayer;
2
3  import modelLayer.parser.*;
4
5  import java.io.IOException;
6
7  import javax.servlet.RequestDispatcher;
8  import javax.servlet.ServletContext;
9  import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 public class Parser extends HttpServlet {
15    private static final long serialVersionUID = 1L;
16
17     public Parser() {
18         super();
19     }
20
21    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
22       System.out.println("in Parser Servelt ...");
23
24       String qsource = request.getParameter("qsource");
25       if(qsource == null) { //The received quiz source was empty.
26          showPreview(request, response, qsource); //It will show the user
               that the quiz was empty.
27          return;
28       }
```

```
29      String error = "";
30      //Split the input on newlines, because the parser runs on
31      //one line at a time.
32      String lines[] = qsource.split("[\\r?\\n]");
33      //String lines[] = qsource.split("[\\n]");
34      QuizScriptParser parser = new QuizScriptParser();
35      int i = 0;
36      while(i < lines.length && error.equals("")) {
37         lines[i]= lines[i]+" <br>";
38         error = parser.run(lines[i]); //parse the line
39         i++;
40      }
41
42      if (error.equals("")) { //got to the end of the input and there was no
            error.
43                        //Now we can insert MMP tags.
44         System.out.println("Parsing done. Now passing to mmp injector");
45         String result = "";
46         QuizScriptMMPInjector pmi = new QuizScriptMMPInjector();
47         i = 0;
48         while(i<lines.length) {
49            result = result + pmi.run(lines[i]);
50            i++;
51         }
52         //System.out.println("Sending to WadgeMMP:\n"+result);
53         //lets try and pass it to Wadge's MMP processor http://dbweb.cs.
               uvic.ca: 8080/servlet/MMPServlet
54         String createdQuiz = passToWadgeMMP(result);
55         //System.out.println("Returned from Wadge MMP:");
56         System.out.print(createdQuiz);
57         System.out.println("Parsing/Injecting DONE.");
58
59         showPreview(request, response, createdQuiz);
60         return;
61
62      }
63      else {
```

```
64          handleError(request, response, i, lines[i-1], error); //-1 because
                we went over by one in the while loop
65          return;
66       }
67
68    }
69
70    private void showPreview(HttpServletRequest request,
71                    HttpServletResponse response,
72                    String createdQuiz) throws ServletException, IOException
                         {
73
74       if(isAjax(request)){
75          createdQuiz = "0 "+ createdQuiz; //0 will be parsed out by js in
                web-browser
76          request.setAttribute("errorExists", false);
77          request.setAttribute("createdQuiz", createdQuiz);
78          passToViewLayer(request, response, "ShowParserResult");
79       }else {
80          request.setAttribute("createdQuiz", createdQuiz);
81          passToViewLayer(request, response, "ShowCreatedQuiz");
82       }
83
84
85
86    }
87    /* Ajax calls set the header "X-Requested-With = XMLHttpReqest"
88     * Further more, because I'm not sure if all browser do that, I send
          extra data
89     * with my ajax request and check them here.
90     * */
91    private boolean isAjax(HttpServletRequest request) {
92       //The standard way
93       if ("XMLHttpRequest".equals(request.getHeader("X-Requested-With"))) {
94          return true;
95       }
96
```

```
97      //If the browser hasn't set the header, look for my flag that was
98      //send with the request.
99      String isAjax = request.getParameter("isAjax");
100     if(isAjax == null || isAjax.equals("")) {
101       return false;
102     }else {
103       return true;
104     }
105
106   }
107
108   private void passToViewLayer(HttpServletRequest request,
          HttpServletResponse response, String whereTo ) throws
          ServletException, IOException {
109     RequestDispatcher dispatcher;
110     ServletContext context = getServletContext();
111     dispatcher = context.getNamedDispatcher(whereTo);
112     if(dispatcher == null) {
113       //log("dispatcher couldn't find the requested servlet: "+name);
114       System.out.println("dispatcher couldn't find the requested servlet:
            "+whereTo);
115
116       request.setAttribute("ErrorMsg", "Could not find the module
            responsible for showing" +
117             "the parser results to the user. ie. 'ShowParserResult'");
118       dispatcher = context.getNamedDispatcher("Error");
119       if(dispatcher == null){
120         System.out.println("Can not find the 'Error.jsp' module."); //
              Nothing else can be done.
121         return;
122       }
123     }
124
125     dispatcher.forward(request, response);
126     return;
127
128   }
```

```
129
130    private String passToWadgeMMP(CharSequence result) {
131       WadgeMMP mmp = new WadgeMMP();
132       return mmp.connect(result);
133
134    }
135
136    protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
137       doPost(request, response);
138    }
139
140    private void handleError(HttpServletRequest request,
141                    HttpServletResponse response,
142                    int lineNum,    //The line number the error is on
143                    String line,    //The line containing the error
144                    String errorMsg ) throws ServletException, IOException {
145
146       request.setAttribute("errorExists",true);
147       request.setAttribute("lineNum", lineNum);
148       request.setAttribute("line", line);
149       request.setAttribute("errorMsg", errorMsg);
150
151       passToViewLayer(request, response,"ShowParserResult");
152
153    }
154 }
```

B.3 Quiz Servlet

```
1  package controllerLayer;
2
3  import java.io.IOException;
4  import java.util.Vector;
5
6  import javax.servlet.RequestDispatcher;
7  import javax.servlet.ServletContext;
```

```
8  import javax.servlet.ServletException;
9  import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 import modelLayer.parser.QuizScriptMMPInjector;
14
15 public class Quiz extends HttpServlet {
16    private static final long serialVersionUID = 1L;
17
18     public Quiz() {
19         super();
20     }
21
22    @SuppressWarnings("unchecked")
23    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
24       System.out.println("in Quiz servlet ...");
25
26       //get the requested quiz id from request
27       String quizID = request.getParameter("id");
28       if(quizID == null || quizID.equals("")) {
29          //== null : to check if the parameter "id" is present or not
30          //.equal("") : to check if the parameter "id" had a value.
31          handleError(request, response, "The quiz ID is null");
32          return;
33       }
34
35
36       //call the db module to retrieve the quiz's source
37       request.setAttribute("quizID", quizID);
38       passToDiffModule(request, response, "GetData", false); //go to model
             layer and come back
39
40       if(request.getAttribute("ErrorMsg") != null) { //nothing was returned
             from database
```

```
41          handleError(request, response, "The requested quiz is not in the
                database.");
42          return;
43       }
44       //ok, so we have data
45
46       Vector<String> sources_vec = (Vector<String>) request.getAttribute("
             sources_vec");
47       String quizSrc = sources_vec.firstElement();
48       System.out.println("quiz source: "+ quizSrc);
49
50       //was it an ajax request?
51       //ajax is used to edit the quiz source in the webpage
52       if(isAjax(request)) {
53          request.setAttribute("errorExists", false);
54          request.setAttribute("createdQuiz", quizSrc);
55          //A little bit of hack. "ShowParserResults" was originally meant to
                only show the parser's results,
56          //but I'm going to use it to send data to the ajax call.
57          passToDiffModule(request, response, "ShowParserResult",true);
58          return;
59       }else{ //the view link was clicked, lets update the view counter in db
60          request.setAttribute("update", true);
61          passToDiffModule(request, response, "GetData", false);
62
63          String errMsg = (String) request.getAttribute("ErrorMsg");
64          if(errMsg != null) { //error returned from db
65             handleError(request, response, errMsg);
66             return;
67          }
68       }
69
70       //inject MMP into quiz
71       QuizScriptMMPInjector pmi = new QuizScriptMMPInjector(quizSrc);
72       StringBuffer quizMMPInjected = pmi.run();
73
74       //send it over to wadge mmp
```

```
75      String lines = quizMMPInjected.toString().replaceAll("[\\r?\\n]", "<br
            >"); //replace the newlines with <br> so it can show on browser
76      String createdQuiz = passToWadgeMMP(lines);
77      //System.out.println(createdQuiz);
78      //TODO:get rid of the ? mark sent by Wadge's mmp.
79
80      //send the created quiz to view layer
81      request.setAttribute("createdQuiz", createdQuiz);
82      passToDiffModule(request, response, "ShowCreatedQuiz",true);
83
84    }
85
86    private void passToDiffModule(HttpServletRequest request,
          HttpServletResponse response, String servletName, boolean isForward )
           throws ServletException, IOException {
87      RequestDispatcher dispatcher;
88      ServletContext context = getServletContext();
89      dispatcher = context.getNamedDispatcher(servletName);
90
91      if(dispatcher == null) {
92        log("Quiz module couldn't find the requested servlet: "+servletName
              );
93
94        request.setAttribute("ErrorMsg", "Could not find the module: "+
              servletName);
95        dispatcher = context.getNamedDispatcher("Error");
96        if(dispatcher == null){
97          System.out.println("Can not find the 'Error.jsp' module."); //
                Nothing else can be done.
98          return;
99        }
100     }
101     if(isForward){
102       dispatcher.forward(request, response);
103     }else {
104       dispatcher.include(request, response);
105     }
```

```
106       return;
107
108    }
109    private void handleError(HttpServletRequest request, HttpServletResponse
           response, String errorMsg) throws ServletException, IOException {
110
111       request.setAttribute("ErrorMsg", errorMsg);
112       passToDiffModule(request, response, "Error", true);
113       return;
114
115    }
116
117    private String passToWadgeMMP(String quizSrc) {
118       WadgeMMP mmp = new WadgeMMP();
119       return mmp.connect(quizSrc);
120
121    }
122
123    protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
124       doPost(request, response);
125    }
126
127    /* Ajax calls set the header "X-Requested-With = XMLHttpReqest"
128     * Further more, because I'm not sure if all browser do that, I send
           extra data
129     * with my ajax request and check them here.
130     * */
131    private boolean isAjax(HttpServletRequest request) {
132       //The standard way
133       if ("XMLHttpRequest".equals(request.getHeader("X-Requested-With"))) {
134          return true;
135       }
136
137       //If the browser hasn't set the header, look for my flag that was
138       //send with the request.
139       String isAjax = request.getParameter("isAjax");
```

```
140       if(isAjax == null || isAjax.equals("")) {
141          return false;
142       }else {
143          return true;
144       }
145
146    }
147
148 }
```

B.4 Save Quiz Servlet

```
 1 package controllerLayer;
 2
 3 import java.io.IOException;
 4
 5 import javax.servlet.RequestDispatcher;
 6 import javax.servlet.ServletContext;
 7 import javax.servlet.ServletException;
 8 import javax.servlet.http.HttpServlet;
 9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class SaveQuiz extends HttpServlet {
13    private static final long serialVersionUID = 1L;
14    //The limits on the maximum number of characters in database table's
          attributes.
15    private final int NAME_LNG = 100; //author's name
16    private final int DESC_LNG = 100; //quiz description
17    private final int SOURCE_LNG = 5000; //size of quiz
18
19      public SaveQuiz() {
20         super();
21     }
22
23    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
```

```
24      System.out.println("in SaveQuiz ...");
25
26      String authorName = request.getParameter("name");
27      String quizDesc = request.getParameter("desc");
28      String quizSrc = request.getParameter("src");
29
30      if(isEmptyArgs(request, response, authorName, quizDesc, quizSrc)){
31         return;
32      }
33      if(isDataTooLarge (request, response, authorName, quizDesc, quizSrc))
           {
34         return;
35      }
36      //no error found. On to saving the quiz in the db.
37      request.setAttribute("authorName", authorName);
38      request.setAttribute("quizDesc", quizDesc);
39      request.setAttribute("quizSrc", quizSrc);
40
41      passToDiffModule(request, response, "SaveData",false); //go to db
           module and come back
42      String errorMsg = (String) request.getAttribute("errorMsg");
43      if(errorMsg == null || errorMsg.equals("")) {
44         int newQuizID = (Integer) request.getAttribute("createdQuizID");
45         String link = newQuizLink(newQuizID); //0 will be parsed out by js
              in web-browser
46         request.setAttribute("errorExists", false);
47         request.setAttribute("createdQuiz", link);
48         passToDiffModule(request, response, "ShowParserResult", true);
49
50      }else {//we had a problem
51         handleError(request, response, errorMsg);
52      }
53   }
54
55   private boolean isDataTooLarge(HttpServletRequest request,
56         HttpServletResponse response, String authorName, String quizDesc,
57         String quizSrc) throws ServletException, IOException {
```

```
58
59      if(authorName.length() > NAME_LNG) {
60         handleError(request, response, "Error. The maximum number of
               characters for the author's name is "+NAME_LNG+".");
61         return true;
62      }
63      if(quizDesc.length() > DESC_LNG) {
64         handleError(request, response, "Error. The maximum number of
               characters for the description is "+DESC_LNG+".");
65         return true;
66      }
67      if(quizSrc.length() > SOURCE_LNG) {
68         handleError(request, response, "Error. The maximum size of quiz
               source is "+SOURCE_LNG+" characters.");
69         return true;
70      }
71
72      return false;
73   }
74
75   private boolean isEmptyArgs(HttpServletRequest request,
       HttpServletResponse response,String authorName, String quizDesc,
       String quizSrc) throws ServletException, IOException {
76      if(authorName == null || authorName.equals("")) {
77         handleError(request, response, "Error. Please provide the quiz
               author's name.");
78         return true;
79      }
80      if(quizDesc == null || quizDesc.equals("")) {
81         handleError(request, response, "Error. Please provide a short
               description of the quiz.");
82         return true;
83      }
84      if(quizSrc == null || quizSrc.equals("")) {
85         handleError(request, response, "Error. Please provide the content
               of the quiz.");
86         return true;
```

```
87      }
88      return false; //The arguments were not empty
89    }
90
91    private void handleError(HttpServletRequest request, HttpServletResponse
          response, String errorMsg) throws ServletException, IOException {
92
93      request.setAttribute("errorMsg", errorMsg);
94      request.setAttribute("generalError", errorMsg);
95      passToDiffModule(request, response, "ShowParserResult", true);
96      return;
97
98    }
99
100   private void passToDiffModule(HttpServletRequest request,
          HttpServletResponse response, String servletName, boolean isForward)
          throws ServletException, IOException {
101     RequestDispatcher dispatcher;
102     ServletContext context = getServletContext();
103     dispatcher = context.getNamedDispatcher(servletName);
104     if(dispatcher == null) {
105       log("Quiz module couldn't find the requested servlet: "+servletName
              );
106
107       request.setAttribute("ErrorMsg", "Could not find the module: "+
              servletName);
108       dispatcher = context.getNamedDispatcher("Error");
109       if(dispatcher == null){
110         System.out.println("Can not find the 'Error.jsp' module."); //
              Nothing else can be done.
111         return;
112       }
113     }
114
115     if(isForward) {
116       dispatcher.forward(request, response);
117     }else {
```

```
118          dispatcher.include(request, response);
119       }
120
121       return;
122
123    }
124
125    private String newQuizLink(int id) {
126       //String baseURL = "http://localhost:8080/quizscript2/quiz?id=";
127       String baseURL = "dbweb.cs.uvic.ca:8080/quizscript2/quiz?id=";
128       String htmlCode = "Your quiz's URL: <textarea rows=\"1\" cols=\"50\">"
              + baseURL+id +"</textarea>";
129       return htmlCode;
130    }
131
132    protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
133       doPost(request, response);
134    }
135
136 }
```

B.5 Wadge MMP Connector Servlet

```
1  package controllerLayer;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.io.InputStreamReader;
7  import java.io.OutputStream;
8  import java.net.URL;
9  import java.net.URLConnection;
10 import java.net.URLEncoder;
11
12 public class WadgeMMP {
13
```

```
14    /* Connects to Wadge's MMP. Passes on the quiz source(marked-up text)
15     * and the required mmp file.
16     * Returns: the created quiz.
17     * */
18    public String connect(String quizSrc) {
19       System.out.println("Lets connect to wadge's mmp program.");
20       String url = "http://dbweb.cs.uvic.ca:8080/servlet/MMPServlet";
21       String charset = "UTF-8";
22       String param1 = quizSrc;
23       String param2 = "yasser-quiz.mmp";
24
25       OutputStream output = null;
26       BufferedReader input = null;
27       InputStream response = null;
28
29       String createdQuiz = ""; //Stores the created quiz returned by Wadge's
             MMP.
30
31       try {
32          String query = String.format("qsource=%s&filename=%s",
33                 URLEncoder.encode(param1, charset),
34                 URLEncoder.encode(param2, charset));
35
36          URLConnection connection = new URL(url).openConnection();
37          connection.setDoOutput(true); //force POST
38          connection.setRequestProperty("Accept-Charset", charset);
39          output = connection.getOutputStream();
40          output.write(query.getBytes(charset));
41
42          response = connection.getInputStream();
43          input = new BufferedReader(new InputStreamReader(response, charset)
               );
44
45          String line;
46          while( (line = input.readLine() ) != null) {
47             createdQuiz += line;
48             createdQuiz +="\n";
```

```
49          }
50       }catch (IOException e) {
51          log("Failed to connect to Wadge's MMP. "+e);
52       }catch(Exception e) {
53          log("General exception connecting to Wadge's MMP. "+e);
54       }finally { //closing time
55          if (input != null) { try { input.close(); }catch (IOException e) {
56             log("Failed to close the input stream after connecting to Wadge'
                   s MMP. "+e); }
57          }
58          if (response != null) { try { response.close(); }catch (IOException
                e) {
59             log("Failed to close the input stream after connecting to Wadge'
                   s MMP. "+e); }
60          }
61          if (output != null) { try { output.close(); }catch (IOException e)
               {
62             log("Failed to close the output stream after connecting to Wadge
                   's MMP. "+e); }
63          }
64       }
65
66       return createdQuiz;
67    }
68
69    public String connect(CharSequence quizSrc) {
70       return connect(quizSrc.toString());
71    }
72
73    private void log(String s) {
74       System.out.println(s);
75    }
76
77 }
```

B.6 UTF Corrector Servlet

```
1  package controllerLayer;
2
3  import java.io.IOException;
4  import javax.servlet.Filter;
5  import javax.servlet.FilterChain;
6  import javax.servlet.FilterConfig;
7  import javax.servlet.ServletException;
8  import javax.servlet.ServletRequest;
9  import javax.servlet.ServletResponse;
10 /*
11  * Set the character encoding for each request to UTF-8.*/
12 public class UTF_corrector implements Filter {
13
14     public UTF_corrector() {
15     }
16    public void destroy() {
17    }
18
19    public void doFilter(ServletRequest request, ServletResponse response,
          FilterChain chain) throws IOException, ServletException {
20       System.out.println("Filter...");
21       if (request.getCharacterEncoding() == null) {
22          request.setCharacterEncoding("UTF-8");
23       }
24       // pass the request along the filter chain
25       chain.doFilter(request, response);
26    }
27
28    public void init(FilterConfig fConfig) throws ServletException {
29    }
30
31 }
```

Appendix C

Implementation of View Layer and Dispatcher

A combination of HTML, JSP and Java make up the View Layer. The JavaScript, jQuery and CSS code directly not implemented by me have been omitted in this section as they can easily be downloaded from their corresponding websites.

C.1 Showing Errors

```
1  package viewLayer;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServlet;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 public class ShowParserResult extends HttpServlet {
12    private static final long serialVersionUID = 1L;
13
14     public ShowParserResult() {
15         super();
16     }
```

```
17
18    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
19       response.setContentType("text/html; charset=UTF-8");
20       response.setCharacterEncoding("UTF-8"); //Redundant as set by
             setcontenttype
21       PrintWriter out = response.getWriter();
22
23       String isGeneralError = (String) request.getAttribute("generalError");
24       if(isGeneralError != null){
25          showGeneralError(out, request);
26          out.close();
27          return;
28       }
29
30       boolean errorExists = (Boolean) request.getAttribute("errorExists");
31       if( errorExists ) {
32          showError(out, request);
33
34       }else{
35          showPreview(out, request);
36       }
37
38       //closing time
39       out.close();
40       return;
41
42
43    }
44
45    private void showPreview(PrintWriter out, HttpServletRequest request) {
46       String createdQuiz = (String) request.getAttribute("createdQuiz");
47       if(createdQuiz == null || createdQuiz.equals("")) {
48          out.println("0 "); //will be parsed out by js on browser
49          out.println("Nothing to show. Empty quiz was submitted");
50          return;
51       }
```

```
52       out.println(createdQuiz); //Show the quiz
53    }
54
55    private void showError(PrintWriter out, HttpServletRequest request) {
56       int lineNum = (Integer) request.getAttribute("lineNum");
57       String line = (String) request.getAttribute("line");
58       String errorMsg = (String) request.getAttribute("errorMsg");
59
60       out.println(lineNum); // will be parsed out by js in web-browser
61       out.println("Error at line: "+lineNum+"<BR>");
62       out.println(line+"<BR>");
63       out.println(errorMsg+"<BR>");
64
65    }
66
67    private void showGeneralError(PrintWriter out, HttpServletRequest
          request) {
68       String errorMsg = (String) request.getAttribute("errorMsg");
69       out.println(errorMsg);
70       System.out.println("Sending: "+errorMsg);
71
72    }
73
74    protected void doGet(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
75       doPost(request, response);
76    }
77 }
```

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.
       w3.org/TR/html4/loose.dtd">
4  <html>
5  <head>
6  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7  <title>Error</title>
```

```
8  </head>
9  <body>
10    There was an error:
11    <br>
12    <%
13       System.out.println("Error jsp page");
14       String ErrorMsg = (String) request.getAttribute("ErrorMsg");
15       if (ErrorMsg != null) {
16          out.write(ErrorMsg);
17       }
18    %>
19 </body>
20 </html>
```

C.2 Home Page

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
       pageEncoding="UTF-8"%>
2  <%@ page import="java.util.Vector"%>
3  <%@ page import="java.util.ArrayList"%>
4
5  <!DOCTYPE html>
6  <html>
7  <head>
8  <meta http-equiv = "Content-type" content = "text/html;charset=UTF-8">
9  <link rel = "stylesheet" href= "http://code.jquery.com/ui/1.10.3/themes/
       smoothness/jquery-ui.css" />
10 <script type ="text/javascript" src="js/jquery-1.9.1.js"></script>
11 <script src = "http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
12
13 <script type = "text/javascript" src = "js/jquery-linedtextarea.js"></
       script>
14 <link type="text/css" href = "css/jquery-linedtextarea.css" rel = "
       stylesheet" />
15
16 <script>
17    var errorLineNum = 0; //The line number with an error on it
```

```
18
19    jQuery(document).ready(function() { // When the HTML DOM is ready
          loading, then execute the following function...
20
21       jQuery("#shareButton , #submitButton").attr("disabled","disabled");
22
23       //Disable the submit and share button if there are any changes to the
             textarea's contents
24       jQuery ("#textareaId").bind ("input propertychange", function (){
25          jQuery ("#shareButton , #submitButton").attr ("disabled","disabled
                ");
26       }); //textarea bind
27
28       //The dialog that pops up to ask user for quiz info before
29       //sending to server to be saved.
30       $( "#dialog-form" ).dialog ({
31           autoOpen: false,
32           height: 300,
33           width: 350,
34           modal: true,
35           buttons: {
36             "Save to data-base": function () {
37
38                //user entered data
39                var name = jQuery("#name").val(); //from dialog
40                var desc = jQuery("#desc").val(); //from dialog
41                var src = jQuery("#textareaId").val(); //from textarea
42                saveToDB(name, desc, src);
43
44                 $( this ).dialog( "close" );
45
46              },
47               Cancel: function() {
48                 $( this ).dialog( "close" );
49               }
50           },
51           close: function() {
```

```
52          //allFields.val( "" ).removeClass( "ui-state-error" );
53          }
54          });//dialog
55
56      jQuery("#shareButton").click(function() {
57         jQuery("#dialog-form").dialog("open");
58      });
59
60      //Here is the AJAX
61      jQuery('#previewButton').click(
62           function(){
63              var ajax = jQuery.post(
64                 "parser", //servlet url
65                 {//data
66                    qsource : jQuery("#textareaId").val(),
67                    isAjax: "true",
68                 } );
69              ajax.done(function (responseText) { //the call back function
                    for ajax
70                 //parses the returned string and converts the first part of
                       it
71                 //to integer. The number represets the line number with the
                       error on it.
72
73                 errorLineNum = 0;
74                 errorLineNum = parseInt(
75                      responseText,
76                      10); //Backward compatiblity
77                 selectLine(errorLineNum);
78                 if(errorLineNum == 0) {
79                    jQuery("#shareButton , #submitButton" ).removeAttr("
                         disabled");
80                 }
81                 var errMsg = responseText.substring( responseText.indexOf("
                       ")); //get rid of the number at the begining of
                      message.
82
```

```
83                jQuery("#preview").empty().append(errMsg); //Put the
                      returned result in the preview section
84             });
85          }); //previewButton.click
86
87    }); //jQuery.ready
88
89    function getSource(quizID) {
90
91       var ajax = jQuery.post(
92          "quiz", //servlet url
93            {//data
94             isAjax : "true",
95             id : quizID,
96            } );
97          ajax.done(function(responseText){
98             document.getElementById('textareaId').value = responseText;
99             //Disable the share and submit button as we just changed the
100            //contents of textarea.
101            jQuery("#textareaId").trigger("input", "propertychange");
102
103         });
104   };
105
106   function saveToDB(authorName, quizDesc, quizSrc) {
107      var ajax = jQuery.post(
108         "SaveQuiz",
109         {
110            name : authorName,
111            desc : quizDesc,
112            src : quizSrc,
113            isAjax : "true",
114         });
115      ajax.done(function (responseText) {
116         jQuery("#preview").empty().append(responseText); //Put the returned
               result in the preview section
117
```

```
118         return true;
119       });
120       return false; //Error, couldn't save the quiz
121    }
122
123    function resizeFrame(id, width, heigth) {
124
125       document.getElementById(id).height = (heigth) + "px";
126       document.getElementById(id).width = (width) + "px";
127    }
128
129    /*Lined text area*/
130    $(function() {
131       $("#textareaId").linedtextarea({
132          selectedLine : errorLineNum
133       });
134    });
135
136    /*
137     * Helper function to change the highlighted error line number in the
138     * margin of the textarea.
139     * Thanks to: http://stackoverflow.com/questions/15868121*/
140    function selectLine(n) {
141       if (n < 0)
142          return false;
143       $(".codelines .lineno.lineselect").removeClass("lineselect");
144       $(".codelines .lineno").eq(n - 1).addClass("lineselect");
145    }
146 </script>
147
148
149
150
151 <style type="text/css">
152 body {
153    background-color: #99CCCC;
154    margin-left: 45px;
```

```
155 }
156 /*p {
157   white-space: pre;
158 }*/
159
160 #left {
161    float: left;
162    width: 750px;
163    height: 100%;
164    overflow: auto;
165 }
166
167 #right {
168    width: 500px;
169    overflow: auto;
170 }
171 #right table tbody {
172    background-color:#999976;
173 }
174
175 #right table thead {
176    background-color:#999966;
177 }
178
179 #tutorialSection {
180    width: 800px;
181    overflow: auto;
182 }
183
184 #tutorialSection table tbody {
185    background-color:#999976;
186 }
187
188 #tutorialSection table thead {
189    background-color:#999966;
190 }
191 .clear {
```

```
192    clear: both;
193 }
194 </style>
195
196 <title>Welcome to QuizScript</title>
197 </head>
198
199 <body>
200    <div id="container">
201       <div id="left">
202          <h2>QuizSript Entry Page</h2>
203          <p>
204          Enter the QuizScript source into the text area and press "Check for
                Error". A preview of the quiz
205          will appear in the "Preview Section". Submitting the quiz will let
               you try it out.
206          Clicking "Share" will store the quiz in database and will provide
               you with a unique URL to it.
207          <p>
208
209          <form action="/quizscript2/parser" method="POST" enctype="
                application/x-www-form-urlencoded; charset=utf-8">
210             <textarea id="textareaId" name="qsource" rows="12" cols="100"
211                placeholder="Start typing quiz here"></textarea>
212
213             <input type="hidden" name="filename" value="quiz.mmp">
214             <button type="button" value="Preview" id="previewButton">Check
                   for Errors</button>
215             <input type="submit" value="Submit" id="submitButton"/>
216             <button type="button" value="Share" id="shareButton">Share</
                   button>
217          </form>
218
219       </div>
220       <!-- End left panel -->
221       <div id="right">
222       <br><br><br><br><br><br><br><br>
```

```
223         <%
224         //the database module puts the information in these Vectors.
225         Vector<String> id_vec = (Vector<String>) request.getAttribute("
                id_vec");
226         Vector<String> desc_vec = (Vector<String>) request.getAttribute("
                desc_vec");
227         Vector<String> views_vec = (Vector<String>) request.getAttribute("
                views_vec");
228         Vector<String> dates_vec = (Vector<String>) request.getAttribute("
                dates_vec");
229         //Vector<String> sources_vec = (Vector<String>) request.
                getAttribute("sources_vec");
230         %>
231         <table border=1 cellspacing=1 cellpadding=5 frame=box
232           bgcolor="#999966">
233           <thead>
234             <tr>
235               <th>ID</th>
236               <th>Description</th>
237               <th>Views</th>
238               <th>View/Modify</th>
239               <th>Date Submitted</th>
240               <!-- <th>DEBUG: source code</th> -->
241             </tr>
242           </thead>
243
244           <% for(int i= 0; i< id_vec.size(); i++) { %>
245           <tr>
246             <td><%= id_vec.get(i) %></td>
247             <td><%= desc_vec.get(i) %></td>
248             <td><%= views_vec.get(i) %></td>
249             <td><a href="/quizscript2/quiz?id=<%=id_vec.get(i)%>"> view <
                    /a> /
250             <a id="editLink" onclick="getSource(<%= id_vec.get(i)%> )"
                    href="javascript:void(0);"> edit </a></td>
251             <td><%= dates_vec.get(i) %></td>
252             <!--DEBUG <td> //sources_vec.get(i) %></td> -->
```

```
253           </tr>
254           <% } %>
255        </table>
256        <!-- activate if you want to show a link to the next 5 quizzes
257        If you need to show more than 5 quizzes at a time then change the
               howMany value
258        -->
259          <a href= "/quizscript2/?offset=<%=request.getAttribute("offset")
               %>&howMany=5"> Show 5 more </a>
260
261     </div>
262     <!-- End right panel -->
263
264     <div id="preview" class="clear">
265        ------------------------------------------------<br>
266        Preview Section<br>
267        ------------------------------------------------<br>
268     </div>
269     <div id="tutorialSection" >
270
271     <h3>QuizScript notation summary:</h3>
272
273        <script>
274        //functions used in the notation summary section to make the
               examples 'live'
275        function dt_toggle(t) {
276           if ( t.parentNode.childNodes[3].style.display=='block' ) {
277              t.parentNode.childNodes[3].style.display='none';
278              t.style.color = 'blue';
279           }
280           else {
281              t.parentNode.childNodes[3].style.display='block';
282              t.style.color = 'red';
283           }
284
285        }
286        function dt_toggle(t)
```

```
287         {
288            if ( t.parentNode.childNodes[3].style.display=='block' ) {
289               t.parentNode.childNodes[3].style.display='none';
290               t.style.color = 'blue';
291              }
292            else {
293              t.parentNode.childNodes[3].style.display='block';
294              t.style.color = 'red';
295              }
296          }
297
298         function dt_toggle(t)
299         {
300            if ( t.parentNode.childNodes[3].style.display=='block' )
301               {
302               t.parentNode.childNodes[3].style.display='none';
303               t.style.color = 'blue';
304              }
305               else {
306               t.parentNode.childNodes[3].style.display='block';
307               t.style.color = 'red';
308              }
309         }
310
311         var errcount = 0;
312
313         function checkans(t,a,b)
314         {
315            for( var i=0; i<a.length; i++)
316                if (a[i].toLowerCase() == b.toLowerCase()) {
317                  iae_right(t,a[i]);
318                  return;
319               } ;
320               iae_wrong(t);
321          }
322
323         function iae_wrong(t)
```

```
324         {
325            t.style.backgroundColor='red';
326             errcount++;
327             document.getElementById('ec').innerHTML = ''+errcount;
328         }
329
330         function iae_right(t,a)
331         {
332            tp=t.parentNode.parentNode.parentNode.parentNode;
333            tp.innerHTML=a;
334            tp.style.color='green';
335            tp.style.backgroundColor='white';
336         }
337
338         function correct(t,a) {
339             if ((t.value==a) || (t.value=='?')) {
340               tp = t.parentNode;
341               tp.innerHTML = a;
342               tp.style.color = 'green';
343               tp.style.backgroundColor = 'white';
344            }
345             if( t.value!=a) {
346               errcount++;
347               document.getElementById('ec').innerHTML = ''+errcount;
348             }
349             if( t.value!='?') {
350               t.value='';
351             }
352          }
353         document.getElementById('tc').innerHTML = ''+7;
354         </script>
355        <table border=1 cellspacing=8 cellpadding=5 rules=rows frame=box
              bgcolor="#999966">
356          <thead>
357            <tr>
358            <th>What you want</th>
359            <th>What you write</th>
```

```
360           <th>Result</th>
361           </tr>
362          </thead>
363
364          <tbody>
365          <tr>
366           <td>Fill-in-the-blank questions</td>
367           <td>Il a perdu =ses cl&eacute;s.</td>
368           <td>Il a perdu
369         <span>
370           <input onchange="correct(this,'ses');" name="Nom" value="" size="
                 13" type="text"><sup>1</sup>
371         </span> cl&eacute;s.<br>
372         (Typing "?" into the blank will reveal the answer)
373           </td>
374          </tr>
375
376          <tr>
377           <td>A clue in form of a word which lights up on mouse over</td>
378           <td>*Pierre* a perdu ses cl&eacute;s.</td>
379           <td>
380         <span style="color: black;"
381               onmouseover="this.style.color='blue';this.style.fontSize='
                     larger';"
382               onmouseout="this.style.color='black';
383                    this.style.fontSize=this.parentNode.style.fontSize;"
384               >
385           Pierre
386         </span> a perdu ses cl&eacute;s.
387           </td>
388          </tr>
389
390          <tr>
391           <td>A pop up hint associated with a word or phrase</td>
392           <td>le repas est (d&eacute;licieux|delicious).</td>
393           <td>le repas est
394         <span style="position: relative; "
```

```
395             onMouseOver="this.childNodes[1].style.opacity=1;"
396             onMouseOut="this.childNodes[1].style.opacity=0;">
397
398              <span  onMouseOver = "event.stopPropagation();"
399                    style="background-color:white; color:blue; opacity: 0;
400             position: absolute; bottom:15px;right: 4px; "
401                    > delicious 
402              </span>d&eacute;licieux.
403           </td>
404         </tr>
405
406         <tr>
407           <td>Multiple choice questions from a particular list</td>
408           <td>Il a perdu [son_cl&eacute; @ses_cl&eacute;s].</td>
409           <td>
410         Il a perdu
411         <div style="position: static; display: inline;"
412             onmouseover="this.style.position='relative';
413                this.childNodes[0].style.display='inline';"
414             onmouseout="this.style.position='static';
415               this.childNodes[0].style.display='none'"
416             ><table style="background-color: white; display: none;
                   position: absolute; top: 15px; z-index: 10;">
417            <tbody style="background-color:white">
418              <tr>
419           <td onclick="checkans(this,['ses cl&eacute;s'],'son cl&eacute;')
                   ">son cl&eacute;</td>
420              </tr>
421              <tr>
422           <td onclick="checkans(this,['ses cl&eacute;s'],'ses cl&eacute;s
                   ')">ses cl&eacute;s</td>
423              </tr>
424            </tbody>
425          </table>__<sup>2</sup></div>.
426
427           </td>
428         </tr>
```

```
429
430          <tr>
431            <td>Specify a list of choices for multiple questions to be used
                   through out the quiz</td>
432            <td>.c mon ma mes<br>
433                .c1 son sa ses</td>
434            <td><!-- nothing goes here--></td>
435          </tr>
436
437          <tr>
438            <td>A multiple choice question from the global list.</td>
439            <td>J'ai perdu #mes cl&eacute;s.<br>
440          Il a perdu 1#ses cl&eacute;s. </td>
441            <td>J'ai perdu
442          <div style="position: static; display: inline;"
443               onmouseover="this.style.position='relative';
444                  this.childNodes[0].style.display='inline';"
445               onmouseout="this.style.position='static';
446                 this.childNodes[0].style.display='none'"><table style="
                      background-color: white; display: none; position:
                      absolute; top: 15px; z-index: 10;">
447              <tbody style="background-color:white">
448                <tr>
449            <td onclick="checkans(this,['mes'],'mon')">mon</td>
450                </tr>
451                <tr>
452            <td onclick="checkans(this,['mes'],'ma')">ma</td>
453                </tr>
454                <tr>
455            <td onclick="checkans(this,['mes'],'mes')">mes</td>
456                </tr>
457              </tbody>
458           </table>__<sup>3</sup></div> cl&eacute;s.<br>
459
460          Il a perdu <div style="position: static; display: inline;"
461               onmouseover="this.style.position='relative';
462                      this.childNodes[0].style.display='inline';"
```

```
463               onmouseout="this.style.position='static';
464                     this.childNodes[0].style.display='none'"><table style="
                          background-color: white; display: none; position:
                          absolute; top: 15px; z-index: 10;">
465             <tbody style="background-color:white">
466               <tr>
467             <td onclick="checkans(this,['ses'],'son')">son</td>
468               </tr>
469               <tr>
470             <td onclick="checkans(this,['ses'],'sa')">sa</td>
471               </tr>
472               <tr>
473             <td onclick="checkans(this,['ses'],'ses')">ses</td>
474               </tr>
475             </tbody>
476           </table>__<sup>4</sup></div> cl&eacute;s.
477          </tr>
478
479          <tbody>
480
481        </table>
482
483 <p><p style="display:inline">
484 <p><p style="display:inline">
485
486  <br>
487  <br>
488  <br>
489  <br>
490  <br>
491
492 <script>document.getElementById('tc').innerHTML = ''+7;</script>
493
494
495       </div>
496       <div id="dialog-form" title="Enter details for the new quiz">
497          <p class="validateTips">All form fields are required.</p>
```

```
498         <form>
499         <fieldset>
500         <label for="name">Author's name:</label>
501         <input type="text" name="name" id="name" placeholder="Your full
               name"/>
502         <label for="desc">Description:</label>
503         <input type="text" name="desc" id="desc" placeholder="Short
               description of quiz" />
504         </fieldset>
505         </form>
506       </div>
507    </div>
508    <!-- container -->
509 </body>
510 </html>
511 <!-- Last modified on Dec. 2013 by Yasser -->
```

C.3 Created Quiz

```
 1 <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
 2 <%@ page import="java.util.*" %>
 3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.
      w3.org/TR/html4/loose.dtd">
 4 <html>
 5 <head>
 6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 7 <title>Quiz Page</title>
 8
 9 </head>
10 <body
11    style="margin-left: 45px; width: 640px; border-style: double; border-
         width: medium; padding: 8px; line-height: 175%; font-size: large;
         background-color: #EEEEFF; font: comic-sans;">
12
13    <%
14       String createdQuiz = (String) request.getAttribute("createdQuiz");
```

```
15       if (createdQuiz == null || createdQuiz.equals("")) {
16          out.println("0 "); //will be parsed out by js on browser
17          out.println("Nothing to show. Empty quiz was submitted");
18          return;
19       }
20       out.println(createdQuiz); //Show the quiz
21    %>
22
23 </body>
24 </html>
```

Appendix D

Implementation of the Dispatcher Layer

```
1  package controllerLayer;
2
3  import java.io.IOException;
4
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletContext;
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class DispatcherServlet extends HttpServlet {
13    private static final long serialVersionUID = 1L;
14    private static final String webAppURI= "/quizscript2";
15
16     public DispatcherServlet() {
17         super();
18     }
19
20    protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
21
```

```
22      RequestDispatcher dispatcher;
23      ServletContext context = getServletContext();
24      /*don't use pathinfo, it doesn't pass the requested url in the
           dispatcher.
25       * use geturi or url then parse it and pass it to controller*/
26      String controllerServlet = "";
27      String uri = request.getRequestURI();
28      StringBuffer url = request.getRequestURL();
29      System.out.println("********************");
30      System.out.println("Dispatcher received: ");
31      System.out.println(" url: "+url);
32
33      if(uri.equals(webAppURI+"/")) { //home page
34         controllerServlet = "HomePageServlet";
35      }else if(uri.contains(webAppURI+"/quiz")) {
36         controllerServlet = "Quiz";
37      }else if(uri.contains(webAppURI+"/parser")) {
38         controllerServlet = "Parser";
39      }else if(uri.contains(webAppURI+"/SaveQuiz")) {
40         controllerServlet = "SaveQuiz";
41      }else {
42         request.setAttribute("ErrorMsg", "The requested servlet is not
              valid");
43         controllerServlet = "Error";
44      }
45
46      System.out.println("Disaptcher sending to: "+controllerServlet);
47      dispatcher = context.getNamedDispatcher(controllerServlet);
48      if(dispatcher == null) {
49         System.out.println("Dispatcher couldn't find the requested servlet:
               "+controllerServlet);
50      }
51
52      dispatcher.forward(request, response);
53      return;
54
55   }
```

```
56
57    protected void doGet(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
58       doPost(request, response); //treat GET the same as POST method
59    }
60
61 }
```

Appendix E

Server's XML Files

The XML file used to configure the web-server is given in Appendix E.1. The XML file responsible for configuring the database connection pool is provided in E.2.

E.1 Complete web.xml File

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:
       //java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/
       javaee/web-app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/
       javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID
       " version="2.5">
3    <display-name>quizscript2</display-name>
4    <welcome-file-list>
5      <welcome-file>Dispatcher</welcome-file>
6    </welcome-file-list>
7    <resource-ref>
8      <description>Creates a pool of connections for the MySQL database</
           description>
9      <res-ref-name>jdbc/quizscript</res-ref-name>
10     <res-type>javax.sql.DataSource</res-type>
11     <res-auth>Container</res-auth>
12   </resource-ref>
13   <servlet>
```

```
14    <description>Dispatches the client's request into the appropriate
          resources, (jsp/servlet/model)</description>
15    <display-name>DispatcherServlet</display-name>
16    <servlet-name>DispatcherServlet</servlet-name>
17    <servlet-class>controllerLayer.DispatcherServlet</servlet-class>
18  </servlet>
19  <servlet-mapping>
20    <servlet-name>DispatcherServlet</servlet-name>
21    <url-pattern>/quiz</url-pattern>
22    <url-pattern>/Dispatcher</url-pattern>
23    <url-pattern>/parser</url-pattern>
24    <url-pattern>/SaveQuiz</url-pattern>
25  </servlet-mapping>
26  <servlet>
27    <display-name>ShowCreatedQuiz</display-name>
28    <servlet-name>ShowCreatedQuiz</servlet-name>
29    <jsp-file>/ShowCreatedQuiz.jsp</jsp-file>
30  </servlet>
31  <servlet>
32    <display-name>HomePage</display-name>
33    <servlet-name>HomePage</servlet-name>
34    <jsp-file>/HomePage.jsp</jsp-file>
35  </servlet>
36  <servlet>
37    <display-name>Error</display-name>
38    <servlet-name>Error</servlet-name>
39    <jsp-file>/Error.jsp</jsp-file>
40  </servlet>
41  <servlet>
42    <description>Gets the appropriate data from db for the homepage</
          description>
43    <display-name>HomePageServlet</display-name>
44    <servlet-name>HomePageServlet</servlet-name>
45    <servlet-class>controllerLayer.HomePageServlet</servlet-class>
46  </servlet>
47  <servlet>
48    <description></description>
```

```
49      <display-name>Test</display-name>
50      <servlet-name>Test</servlet-name>
51      <servlet-class>test.Test</servlet-class>
52    </servlet>
53    <servlet-mapping>
54      <servlet-name>Test</servlet-name>
55      <url-pattern>/Test</url-pattern>
56    </servlet-mapping>
57    <servlet>
58      <description></description>
59      <display-name>GetData</display-name>
60      <servlet-name>GetData</servlet-name>
61      <servlet-class>modelLayer.db.GetData</servlet-class>
62    </servlet>
63    <servlet>
64      <description></description>
65      <display-name>Quiz</display-name>
66      <servlet-name>Quiz</servlet-name>
67      <servlet-class>controllerLayer.Quiz</servlet-class>
68    </servlet>
69    <servlet>
70      <description>The Parser Servlet in Controller Layer</description>
71      <display-name>Parser</display-name>
72      <servlet-name>Parser</servlet-name>
73      <servlet-class>controllerLayer.Parser</servlet-class>
74    </servlet>
75    <servlet>
76      <description>Displays the result of the parser to the user, either
            error or the quiz preview</description>
77      <display-name>ShowParserResult</display-name>
78      <servlet-name>ShowParserResult</servlet-name>
79      <servlet-class>viewLayer.ShowParserResult</servlet-class>
80    </servlet>
81     <filter>
82      <display-name>UTF_corrector</display-name>
83      <filter-name>UTF_corrector</filter-name>
84      <filter-class>controllerLayer.UTF_corrector</filter-class>
```

```
85    </filter>
86    <filter-mapping>
87      <filter-name>UTF_corrector</filter-name>
88      <url-pattern>/*</url-pattern>
89    </filter-mapping>
90    <servlet>
91      <description>Gets the required quiz info and passes on to db module to
            be saved in db.</description>
92      <display-name>SaveQuiz</display-name>
93      <servlet-name>SaveQuiz</servlet-name>
94      <servlet-class>controllerLayer.SaveQuiz</servlet-class>
95    </servlet>
96    <servlet>
97      <description></description>
98      <display-name>SaveData</display-name>
99      <servlet-name>SaveData</servlet-name>
100     <servlet-class>modelLayer.db.SaveData</servlet-class>
101   </servlet>
102 </web-app>
```

E.2 Database Connection Pool

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3
4  <Context>
5    <!-- source: http://tomcat.apache.org/tomcat-6.0-doc/jndi-datasource-
         examples-howto.html
6         -->
7    <!-- auth: Container or Application
8         -->
9    <!-- type: The fully qualified Java class name expected by the web
         application
10        for this environment entry.
11     -->
12     <!-- maxActive: Maximum number of database connections in pool. Make
           sure you
```

```
13          configure your mysqld max_connections large enough to handle
14          all of your db connections. Set to -1 for no limit.
15          -->
16
17      <!-- maxIdle: Maximum number of idle database connections to retain in
            pool.
18          Set to -1 for no limit. See also the DBCP documentation on this
19          and the minEvictableIdleTimeMillis configuration parameter.
20          -->
21
22      <!-- maxWait: Maximum time to wait for a database connection to become
            available
23          in ms, in this example 10 seconds. An Exception is thrown if
24          this timeout is exceeded. Set to -1 to wait indefinitely.
25          -->
26
27      <!-- username and password: MySQL username and password for database
            connections -->
28      <!-- validationQuery: SQL query that can be used by the pool to
            validate connections
29         before they are returned to the application. If the connection is
               closed it will
30         be aborted and new one is created in its place. -->
31
32      <!-- driverClassName: Class name for the old mm.mysql JDBC driver is
33          org.gjt.mm.mysql.Driver - we recommend using Connector/J though.
34          Class name for the official MySQL Connector/J driver is com.mysql.
                jdbc.Driver.
35          -->
36
37      <!-- url: The JDBC connection url for connecting to your MySQL database
            .
38          -->
39
40    <Resource name="jdbc/quizscript"
41          auth="Container"
42          type="javax.sql.DataSource"
```

```
43              maxActive="100"
44              maxIdle="30"
45              maxWait="10000"
46              validationQuery="select 1"
47              username="intentionally not included"
48              password="intentionally not included"
49              driverClassName="com.mysql.jdbc.Driver"
50              url= "jdbc:mysql://127.0.0.1:3306/quizscript?useUnicode=true&amp
                    ;characterEncoding=utf8"/>
51
52  </Context>
```

Bibliography

[1] B. Beatty and C. Ulasewicz, "Faculty Perspectives on Moving from Blackboard to the Moodle Learning Management System," *TechTrends*, vol. 50, no. 4, pp. 36–45, 2006.

[2] S. Graf and B. List, "An evaluation of open source e-learning platforms stressing adaptation issues," in *Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on*, pp. 163–165, 2005.

[3] W. Wadge, "QuizScript Home Page." http://dbweb.cs.uvic.ca:8080/servlet/MMPServlet?filename=quizscript.mmp, Aug. 2013.

[4] B. F. Skinner, "Teaching Machines," *Science*, vol. 128, no. 3330, pp. pp. 969–977, 1958.

[5] mindflash.com, "Learning Management System to Create Online Training Courses." http://www.mindflash.com/learning-management-systems/what-is-lms, Aug. 2013.

[6] S. McNeil, "A Hypertext History of Instructional Design." http://faculty.coe.uh.edu/smcneil/cuin6373/idhistory/pressey.html, Aug. 2013.

[7] K. Noesgaard, "Bridging eLearning and Social Networks," Master's thesis, University of Victoria, 2008.

[8] H. Coates, R. James, and G. Baldwin, "A Critical Examination of the Effects of Learning Management Systems on University Teaching and Learning," *Tertiary Education and Management*, vol. 11, no. 1, pp. 19–36, 2005.

[9] D. Reed, "LMS Strategies in Higher Education." http://elearnmag.acm.org/featured.cfm?aid=1925841, Aug. 2011.

[10] S. Williams van Rooij, "Open-source learning management systems: a predictive model for higher education," *Journal of Computer Assisted Learning*, vol. 28, no. 2, pp. 114–125, 2012.

[11] Ontario's Ministry of Education, "Online Classroom." http://www.edu.gov.on.ca/elearning/courses.html, Aug. 2012.

[12] wikipedia.org, "Learning Management System." http://en.wikipedia.org/wiki/Learning_management_system#cite_note-8, Aug. 2013.

[13] Green, Kenneth C., "Compus Computting Project." http://www.campuscomputing.net/sites/www.campuscomputing.net/files/Green-CampusComputing2011_4.pdf, Aug. 2011.

[14] Moodle 2.5 Documentation, "Moodle Statistics." https://moodle.org/stats/, Aug. 2013.

[15] Moodle 2.5 Documentation, "Moodle Background." http://docs.moodle.org/25/en/Background, Aug. 2013.

[16] i-newswire.com, "Using Moodle: Using the Popular Open Source Course Management System." http://www.i-newswire.com/using-moodle-using-the-popular/a40110, Aug. 2013.

[17] M. Dougiamas and P. Taylor, "Moodle: Using learning communities to create an open source course management system," in *World conference on educational multimedia, hypermedia and telecommunications*, vol. 2003, pp. 171–178, 2003.

[18] Moodle 2.5 Documentation, "Mutliple Choice question type." http://docs.moodle.org/25/en/question/type/multichoice, Aug. 2013.

[19] Moodle 2.5 Documentation, "Embedded Answers (Cloze) question type." http://docs.moodle.org/25/en/question/type/multianswer, Aug. 2013.

[20] W. Wadge, "Quiz Script - interactive excercises for the masses." http://dbweb.cs.uvic.ca:8080/Quizscript.html, Aug. 2013.

[21] W. Wadge, "Correspondence with Dr. Wadge," 2009–2013.

[22] D. Reed, "Apache Tomcat 6.0 Documentation - JNDI Datasource." http://tomcat.apache.org/tomcat-6.0-doc/jndi-datasource-examples-howto.html, Aug. 2013.

[23] A. V. Aho *et al.*, *Compilers: principles, techniques, & tools*. Boston : Pearson-/Addison Wesley, 2007.

[24] D. Galles, *Moderen Compiler Design*. Addison-Wesley, 2005.