CO7201 Individual Project Dissertation

Project title:	ESB Audit Software
Department:	Computer Science
Author:	Zhipeng Liang
Student ID:	129044658(ZL91)
Email:	zl91@student.le.ac.uk
Supervisor:	Dr Irek Ulidowski
Second Marker:	Prof Reiko Heckel
Date of Submission:	May 16, 2014

Abstract

The topic of the individual project is called 'ESB Audit Software'. The highlight of the project is the cooperation with a company named European Safety Bureau[1] (ESB). However the audit project has been implementing by an outsourcing company called Drupal in India before I commence with the work. Alternatively I was asked to manage a new project called 'Inviso Incident' which is another sub-project of ESB web-based application. What I have done is based on the requirement from it.

This individual project combines technical and academic challenges with that of interacting with a real client. There are two aims with the project.

The one is the management of 'Inviso Incident' project. This aim offers helps to ESB in the aspects of development management, project planning, requirements, prototyping and testing.

The other is the development of 'dynamic form' software. As both of 'ESB Audit' software and 'Inviso Incident' have the common requirement about customising various templates of questionnaire and report by user, the development of 'dynamic form' thus aims for offering the functionalities for users to customise templates which are stored and filled up for the business purposes.

The 'dynamic form' software, which is a web-based application based on a full-stack framework called AppFuse[2], has been developed and tested.

Keywords: AppFuse, Dynamic Form, Project Management, Java EE[3], Open Source

---

[1] EUROPEAN SAFETY BUREAU, 2014-last update, The European Safety Bureau. Available: http://www.esb.eu.com/ [4/27, 2014]

[2] RAIBLE, M., 2014-last update, AppFuse. Available: http://appfuse.org/display/APF/Home [04/27, 2014]

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: ZHIPENG LIANG

Signed:

Date:

[3] 2014-last update, Java Platform, Enterprise Edition (Java EE). Available: http://www.oracle.com/technetwork/java/javaee/overview/index.html [5/1/2014, 2014]

Content

Chapter 1 Introduction

The introduction dedicates to the background of the ESB Company and the motivation of 'Inviso Incident' and 'ESB Audit'.

ESB, with which the author cooperated, is an independent consultancy. The services offered by ESB cover fire safety, food safety and health safety to various industries such as the catering, hotel, retail and so on. ESB's clients have to comply with all kinds of legislation and best practice of food safety. The health & safety of customers and employees from the ESB's clients are protected by ESB ensuring that safe system have been checked and qualified with the principles of risk assessment, incident reporting and investigation, and so on.

The original statement of work (SOW) (Project Management Institute 2013: 68) of 'Inviso Incident' is from CH & Co[4] which is one of ESB clients. CH & Co established in 1991 consists of a number of specialist businesses. One of them is to provide commercial catering to a variety of organisations and companies across the UK. It has a health and safety management system for ensuring the wellbeing of its staff, customers and clients. However some limitations within the current incident reporting and investigation process are found. Therefore CH & Co is keen to improve their internal accident reporting and investigation process.

ESB takes the opportunity and develops 'Inviso Incident' for offering an online service of incident reporting and investigation. Once the project gets online, not only can CH & Co improve the efficiency of incident reporting and investigation, but also ESB can expend their market by offering the service to other companies who are running the similar business.

'ESB Audit' software, which is the original topic of the individual project, comes from ESB as well. The aim of 'ESB Audit' software is for developing an audit system which ESB can integrate it with its own business and gain competitive advantage in the

[4] 2014-last update, CH&Co. Available: http://www.chandco.net/ [4/28/2014, 2014]

consultancy marketplace. This is how the business worked at that moment. The blue chip clients based through auditing, training and policy development are supported by ESB who conduct the audits and send the results with the analysed performance figures to clients by hired third party software.

The contents of the rest illustrate the two aims with their objectives and achievements as the following: project aims and objectives, development of dynamic form, management of Inviso Incident and an overall conclusion.

Chapter 2 Aims and Objectives

At first, there are two aims in the individual project. The one is management of 'Inviso Incident' project. The other is development of 'dynamic form' software

The aim of management of 'Inviso Incident' project is for helping ESB in the aspects of development management, project planning, requirements, prototyping and testing. This aim is changed from the original topic of 'ESB Audit Software'. As the client of ESB is keen to get the available audit system online, ESB decided to initialise the project before the author's participation, and shrink the software development time by outsourcing the implementation to a company in India. This company implement the 'ESB Audit Software' based on PHP[5] language. When the author commenced the cooperation with ESB, there was another project called 'Inviso Incident'. The author was asked to take the responsibility of management of 'Inviso Incident', as the responsibility of implementation part was taken by the outsourcing company. The ESB make those decisions for the following reasons:

- 'Inviso Incident' project is a sub-system of ESB online business as the same as 'ESB Audit Software' and 'ESB Audit Software' is in the process of implementation on PHP. For the technic consistent, ESB decide to hire the outsourcing company implement it on PHP as well.
- The author is good at Java programming but have no idea about PHP, so he can little help with the implementation of 'Inviso Incident'.
- The consideration from business, if the author participates in the implementation of 'Inviso Incident', it may bring new problems to the outsourcing team. As a newbie of PHP, he may slow down the process of development.

The idea of 'dynamic form' software is from 'ESB Audit' software and 'Inviso Incident'. The 'ESB Audit' software is the initial requirements of the individual

[5] THE PHP GROUP, 2014-last update, PHP: Hypertext Preprocessor. Available: http://www.php.net/ [4/29/2014, 2014]

project. The requirement is about the development of web-based auditing software which runs on the web in portable and static devices. The aims of the software require the system to have an audit input, where audit templates can be created and content filled out accordingly, a database to store the data, and a management function for both ESB staff and clients to review performance of particular sites. The core functionality of 'Inviso Incident' is to offer nine types of incident report form and seven types of incident investigation form for clients filled out. Both of 'ESB Audit' and 'Inviso Incident' can be extended from 'dynamic form' software. As 'dynamic form' software allows user customising templates according the needs of various businesses. Once the 'dynamic form' software developed, it can meet the core requirements of both 'ESB Audit' and 'Inviso Incident'.

Secondly, the list below represents the objectives which the author has done for achieving the aim of 'dynamic form' software.

- Template management
- Content can be filled out and stored in a database
- Review filled forms and print as reports
- Run on the web in portable devices
- Supporting the key HTML elements
- Testing
- A visualized process of template design

At last, the following briefly illustrates the objectives which the author has done for achieving the aim of managing 'Inviso Incident' project.

- Circumstance investigation
- Scope management
- Time management
- Design of prototyping

Chapter 3 Development of Dynamic Form

3.1 Essential Requirements

There are three essential requirements in the software. All of them are achieved.

Template management

The objective of template management is for offering functionalities to user building up their template for the various businesses. It is, therefore, one of the most important requirements of the software. A template of form may have an uncertain number of questions. For each question, according to the different of the question, there might be an uncertain number of answers. For example, the question, asking for the surname and the forename, has two text fields for user filling out.

Therefore the requirement of template can be elaborated into the following sub-requirements:

- The template management
    - A view of template list
    - Create and edit a template
    - Active and discard a template
- The question management
    - A view of question list
    - Create, edit and delete a question
    - A question may have sub-questions
- The answer management
    - A view of answer list
    - Create, edit and delete an answer
    - A answer may refer to another answer or question
- The assignment between them
    - Assign a question to a template
    - Assign an answer to a question

Content can be filled out and stored in a database

Given a template has been build up, the software allow a user to fill it out and store those inputs into database, which is one of solution of data persistence. Thus a template may be filled by many users, or filled by one user with many times. Note that one user filling a template many times means the user creates many instances of template, and those instances can be updated and deleted respectively. It is different form that a user fill out a template (one instance of template created), and updating it for many times (still updating the same instance of template). The other implicit requirement is that a user can update or delete the filled out forms (instances of template) created by himself. Other users can only view them, as the other users do not own it.

Review filled forms and print as reports

The filled out forms need a good presentation for viewing and printing. It is, therefore, necessary for the application offering a report view for each filled form. After user clicks the print button on the page of report view, the software represents a well formatted report ready for printing.

3.2 Recommended Requirements

There are three recommended requirements for the software. The first of running on the web in portable devices is achieved. The second of supporting key HyperText Markup Language (HTML) elements is achieved. The third of verification for inputs sound a great idea, but it is impractical for both end users and the software reliability.

Run on the web in portable devices

On one hand this web-based software is supposed to run on a portable device for example a tablet, when a user is filling out a form. On the other hand, it should offer powerful functionalities and easy-use user interface when a user is creating and editing a complex template on a traditional web browser like Internet Explorer, Firefox and so on. It means the web application needs to meet the following items.

- Cross platform. Running on both portable devices and PCs.

- Cross browser. Running on the main stream browsers Such as Internet Explorer, Firefox and so on.
- Automatically adapt to various screen size. As the various devices are with the diverse size of screen, appropriately representing the content of pages requires the application adapting the layout to the width of screens.

Supporting the key HTML elements

The template page may consists of many elements of HTML such as radio button, checkbox, text field, text area, file uploading, and selection box, button, hyper link and so on. Therefore the software ideally is supposed to support all of HTML elements. However an implementation with fully supported HTML elements is not necessary. The analysis of requirement illustrates that six key elements are enough for building all kinds of forms. The application supports those key elements of radio button, checkbox, text field, text area, file uploading/download and label.

3.3 Optional Requirements

There are two optional requirements of the software, Testing and visualized process of template design. Both of them are achieved.

Testing

Fully testing all the functions of the software automatically is not only great helpful for verifying the error corrections, but also can find new bugs when fixing the old bugs or introducing a new function. On one hand, the unit testing is necessary for ensuring each module works as expectation. On the other hand, the regression testing is required for preventing from introducing new bugs or errors when fixing bugs or adding new features.

A visualized process of template design

A visualized process of building up templates is required in the software. For implementing the requirement the application is supposed to offer some kind of user interfaces for end users to manipulate those elements in a template. There are two ways for fulfilling this visualized process.

The one is called ‘assign to’. With this way, users build their templates by assign the elements into it. For example, on the page of template edit, a user assigns a text area into the template for collecting the input information about address.

Another is known as ‘drag and drop’. It means that there is a page for user dragging/picking some elements and dropping them into the template.

Comparing those two solutions, the latter is more convenient for users locating and managing the elements within a template. However the former is easier for implementation. In this software, the way of ‘assign to’ meets to this requirement.

3.4 Technical Specifications

The ‘dynamic form’ software is a web-based, portable device friendly and built on the open source framework of AppFuse information system. Those characters determine its implementation combines various open source projects and Java EE specifications.

Information system

As an information system it is capable to create, update template and allow users fill them out and save the information into a database. MySQL, therefore, is chosen as the implementation for the requirement.

Specifications from Java EE

- JavaServer Pages (JSP). It is from Java Specification Requests[6] (JSR) 245. The purpose of introduction is for generating dynamic content on pages. Another java technology Servlet is able to generate output to client as well. However, comparing to JSP, it is difficult to verify the correction of the mark syntax in a Java program, as the Java compiler handle the HTML tags as strings. (Cole, McChesney et al. 2011: 195) The version of JSP introduced into the software is 2.1 implemented by the library file called javax.servlet.jsp:jsp-api:2.1.

[6] THE JAVA COMMUNITY PROCESS(SM) PROGRAM, 2014-last update, JSRs: Java Specification Requests - JSR Overview. Available: https://jcp.org/en/jsr/overview [5/2/2014, 2014]

- Java Persistence API (JPA). It is from JSR 338. 'JPA defines an API for the management of persistence and object/relational mapping using a Java domain model.'(Gupta 2013: Chapter 13). The one reason of introduction is for implementing the requirements of storing data into a database. Another is for sampling the development. As JPA is capable of object/relational mapping and manipulating data with Java entity objects, there is no need for developing the SQL scripts for building tables, creating, updating or deleting records. The version of JPA introduced into the software is 2.0 implemented by the library file called org.hibernate.java-persistence:jpa-api:2.0-cr-1.

Open source libraries, tools and frameworks

- Maven[7]. It, coming from Apache[8], is a management tool for developing software. The purpose of introduction is for building up the environment of development, managing the dependent libraries, compiling codes, running application and testing. The version of Maven introduced into the software is 3.2.1.
- Spring. It is a framework based on Java language with some great features such as dependency injection, Aspect-Oriented Programming and so on. The purpose of introduction is for integrating various open source frameworks, libraries and tools into the software, such as Hibernate, Struts2. 'Spring is a multitier framework that is not dedicated to a particular architecture.'(Seddighi 2009: 7) The version of Spring Core introduced into the software is 4.0 implemented by the library file called org.springframework:spring-core:4.0.0.RELEASE.

[7] APACHE, 2014-last update, Maven. Available: http://maven.apache.org/ [5/14/2014, 2014]

[8] APACHE, 2014-last update, The Apache Software Foundation. Available: http://apache.org/ [5/2/2014, 2014]

- Hibernate[9]. It is a framework aiming to solve the problems of managing data from relational databases in Java. Hibernate implements JPA specification. It means that on one hand, the annotations of the entity classes are based on JPA specification; on the other hand, the persistent layer is supported by the implementation of Hibernate. One of the evidence is the configuration file of Hibernate in the software called hibernate.cfg.xml, which is for the application registering the entity classes to Hibernate. The other evidence is the version of Hibernate Core introduced into the software is 4.2.7 implemented by the library file called Maven: org.hibernate:hibernate-core:4.2.7.Final.
- Struts2[10]. It is a Java web framework from Apache and based on the Java EE specifications and APIs such as Servlet API 2.4 and JSP API 2.0. This framework has been integrated into AppFuse for simplifying the development on the web tier. For example, it offers a feature of saving the uploading file on the server end.
- Object Graph Navigation Library[11] (OGNL) is for Object-Graph Navigation Language which is introduced into the software for manipulating variables in JSP files.
- JavaServer Pages Standard Tag Library[12] (JSTL) is a part of the Java EE specifications. The core and fmt tags are introduced in the JSP for manipulating variables and formatting the outputs in JSP files.

---

[9] HIBERNATE, 2014-last update, Hibernate. Everything data. Available: http://hibernate.org/ [5/2/2014, 2014]

[10] APACHE, 2014-last update, Struts 2. Available: http://struts.apache.org/development/2.x/ [5/9/2014, 2014]

[11] APACHE, 2014-last update, Object Graph Navigation Library. Available: http://commons.apache.org/proper/commons-ognl/ [5/9/2014, 2014]

[12] ORACLE, 2014-last update, JavaServer Pages Standard Tag Library. Available: http://www.oracle.com/technetwork/java/index-jsp-135995.html [5/9/2014, 2014]

- Bootstrap[13] is a front-end framework for implementing the feature of adapting to the portable devices.
- JUnit[14] is a Java testing framework. It is introduced for building up test cases for testing the implementation methods of Data Access Object (DAO), Service and Action.
- Jetty[15] and Tomcat[16] are Java web servers implementing the specifications of Servlet and JSP. Jetty is introduced in the phases of development and testing. Tomcat is introduced in the phase of deployment.

3.5 Challenges

It is a super challenge to develop the software composited so many technologies. The author learned all those technologies above and applied them into the development. The most important concepts and technologies learned during the development are following:

- Inversion of Control (IoC) pattern. Seddighi (2009: 231) states that 'Spring provides IoC at the heart of its framework'. The beans are defined in the file of dynamicform\core\src\main\resources\applicationContext.xml, which represents a way of registering the named object to the IoC container.
- Object-Relational Mapping (ORM). Goncalves (2013: 113) states 'ORM, which is the mechanism to map objects to data stored in a relational database.' As a part of the implementation of application, the annotation of JPA is introduced to define the entities and their relations.

[13] , Bootstrap. Available: http://getbootstrap.com/ [5/9/2014, 2014]

[14] 2014-last update, JUnit. Available: http://junit.org/ [5/1/2014, 2014]

[15] ECLIPSE, 2014-last update, Jetty - Servlet Engine and Http Server. Available: http://www.eclipse.org/jetty/ [4/27/2014, 2014]

[16] APACHE, 2014-last update, Tomcat. Available: http://tomcat.apache.org/ [4/27/2014, 2014]

- Responsive Web Design. 'Flexible designs make no assumptions about a browser window's width, and adapt beautifully to devices that have portrait and landscape modes.' (Marcotte 2014) Bootstrap offers a way of designing a responsive web application which can adapt to a range of width of various devices.

3.6 Design

Feature model

The major elements on the pages in 'dynamic form' software are text input, label, choice and file.

- The text input includes two types of one-line text input and multi-line text input (known as text area).
- The label is for commenting what kind of information should be enter into the text field. In addition, the label is utilised to illustrate the content of questions as well.
- The choice includes two type of single choice (known as radio button) and multi choice (know as checkbox).
- The file includes two type of file upload and file download.

The following sketch represents the featrue model of the dynamic form.

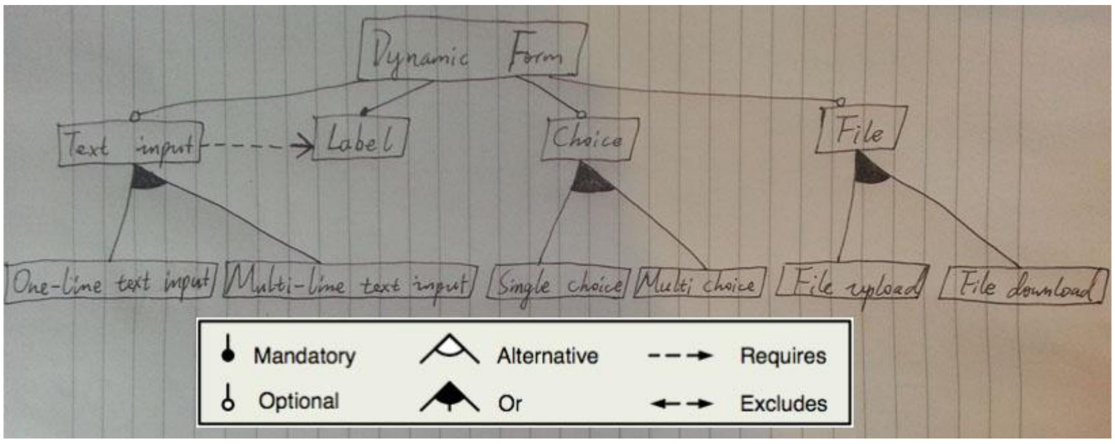
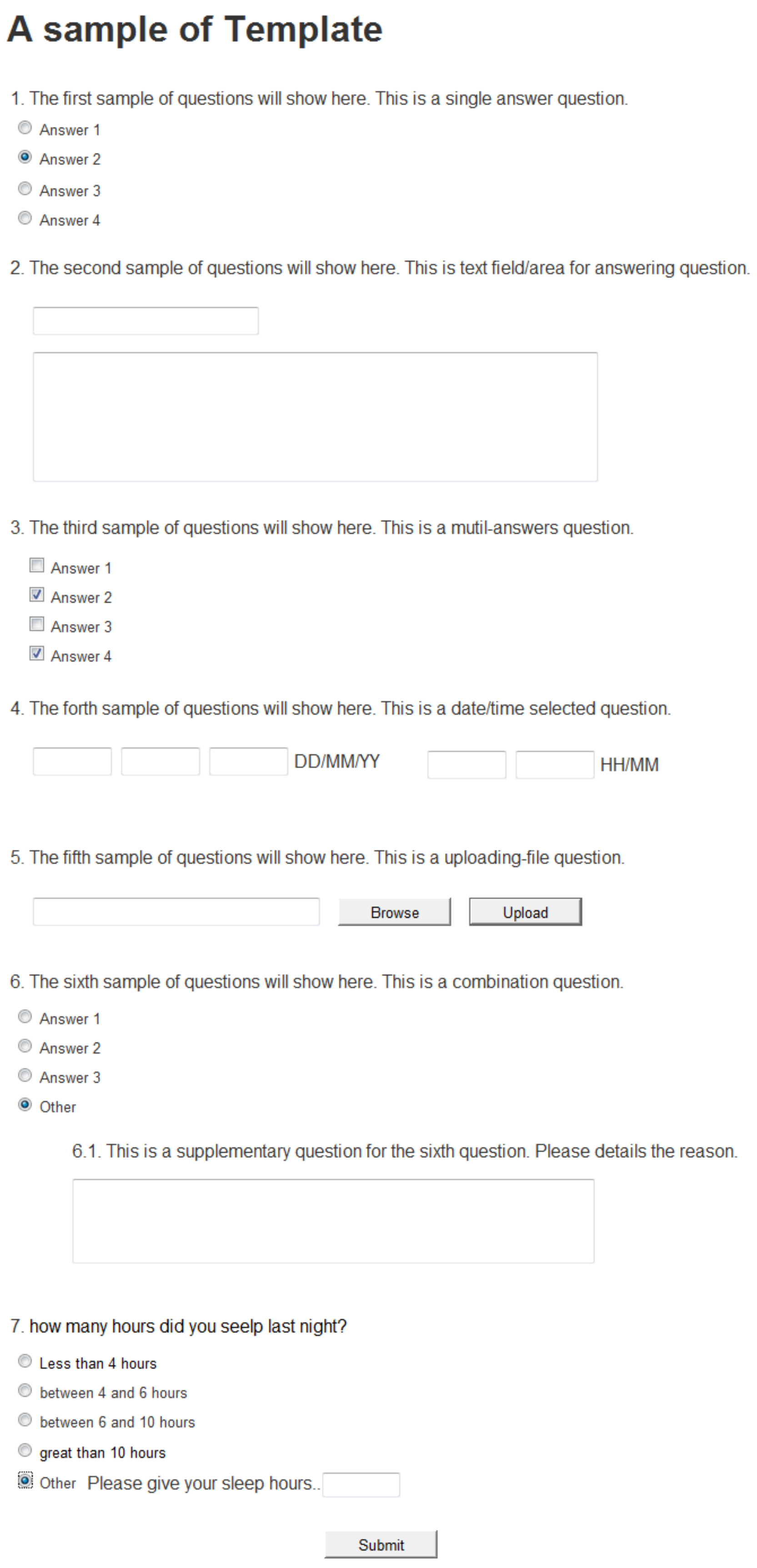


Figure 3-6-1: the featrue model of the dynamic form

Prototyping design

The purpose of prototyping design is for visualising the imaginary user interface. The tool of Axure[17] is introduced for helping the prototyping design. The following is an example of template.

[17] AXURE, 2014-last update, Interactive Wireframe Software & Mockup Tool. Available: http://www.axure.com/ [4/30/2014, 2014]



A sample of Template

1. The first sample of questions will show here. This is a single answer question.

Answer 1
[✓ Answer 2]
Answer 3
Answer 4

2. The second sample of questions will show here. This is text field/area for answering question.

3. The third sample of questions will show here. This is a mutil-answers question.

Answer 1
[✓ Answer 2]
Answer 3
[✓ Answer 4]

4. The forth sample of questions will show here. This is a date/time selected question.

DD/MM/YY HH/MM

5. The fifth sample of questions will show here. This is a uploading-file question.

6. The sixth sample of questions will show here. This is a combination question.

Answer 1
Answer 2
Answer 3
[✓ Other]

6.1. This is a supplementary question for the sixth question. Please details the reason.

7. how many hours did you seelp last night?

Less than 4 hours
between 4 and 6 hours
between 6 and 10 hours
great than 10 hours
[✓ Other] Please give your sleep hours..

Figure 3-6-2: a example of template

Database design

The database design includes two parts. The one is Entity–relationship model (ER model); the other is Physical data model.

The ER model below represents the conceptual data aspects of a dynamic form system.

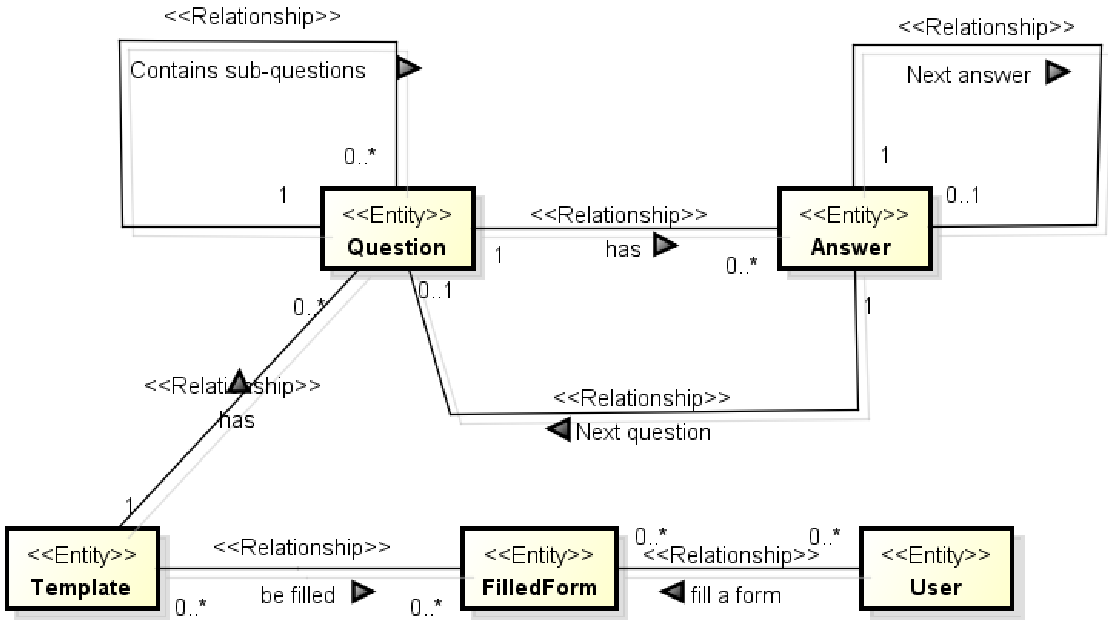


Figure 3-6-3: ER model of dynamic form

- A template has many questions, and each question has many answers.
- A user can fill a template with any times. In reverse, a template can be filled many times by any user. The entity of Filled Form contains the result which a user fills.
- A question could have many sub-questions.
- An answer may refer to another answer or question. The design for relations of ‘Next answer’ and ‘Next question’ is for the reason that another answer or question show up, when the current answer is selected or active. For example, a user select the last answer item of ‘Other’, then the text field with a label of ‘Please specify’ shows up.

The physical data model, refined from ER model, represents the design of a scheme within a relation database.

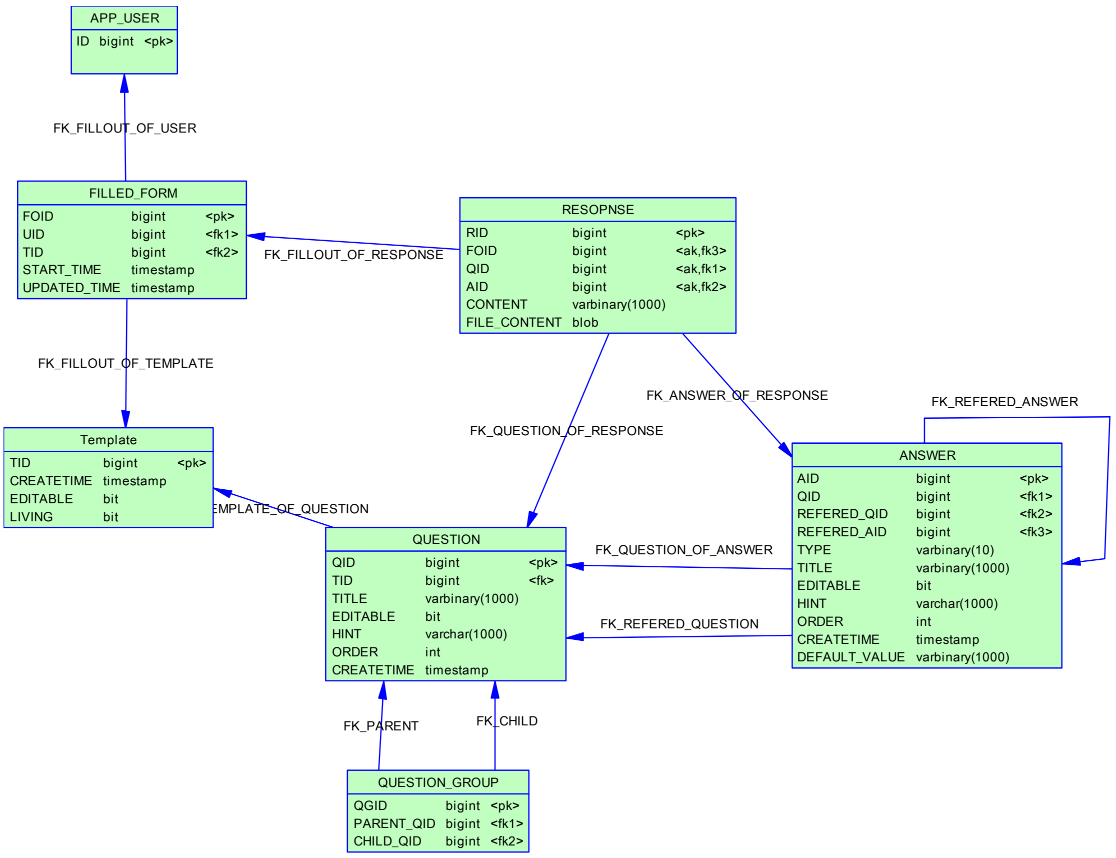


Figure 3-6-4: the physical data model of dynamic form

- The table of app_user represents the user entity in ER model.
- The table of filled form, template, question and answer is responsible to the same name entities in ER model.
- The response table is for keeping the content which a user fills on each answer in a template.
- The table of question group represents the relation between parent question and children questions.

Entity classes design

Five entity classes, defined in the application, are the entities of Template, Question, Answer, FilledForm and Response. The AnswerType and TemplateStatus are classes of Enumeration. The Enumerations are defined by keyword '*enum*' which allow developers create a new type with a restricted set of named values, and treat those values as regular program components.(Bruce Eckel 2006: 725). The diagram of entity classes is below.

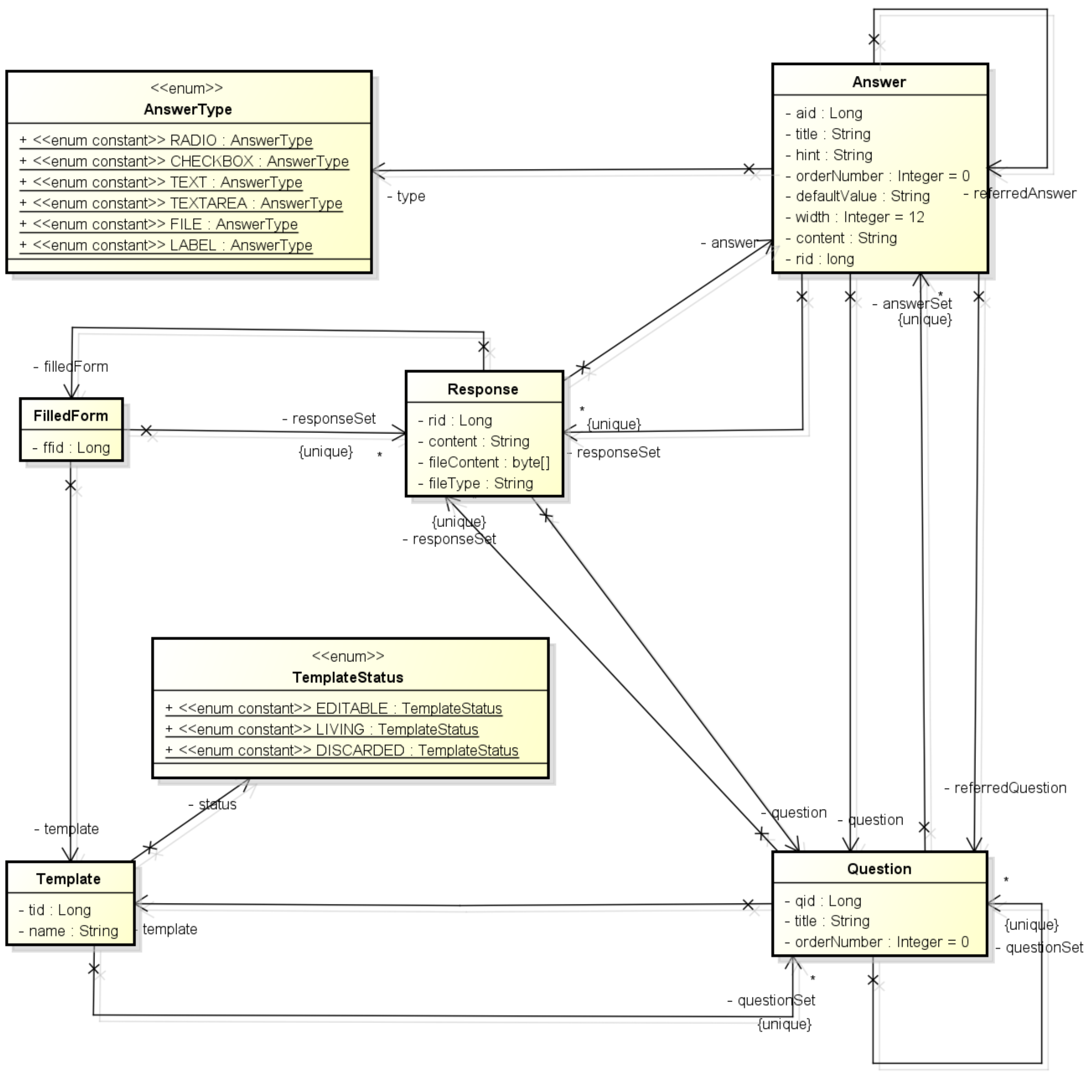


Figure 3-6-5: the enities classes diagram of dynamic form

Architecture design

The architecture design is similar to the paradigm of Java EE multi-tiered applications. There are three tiers in this application. In the client tier, users access Java EE server throughout browser on a PC or portable device. In the Java EE Server, there are two tiers. One is web tier, another is business tier. In the web tier JSPs, instead of JSF, response to users' requests with the dynamic content. In the business tier, Services (instead of Enterprise Beans) deal with complex computations and manipulate entity objects via JPA. In the Enterprise Information System (EIS) tier, information is stored into database servers.

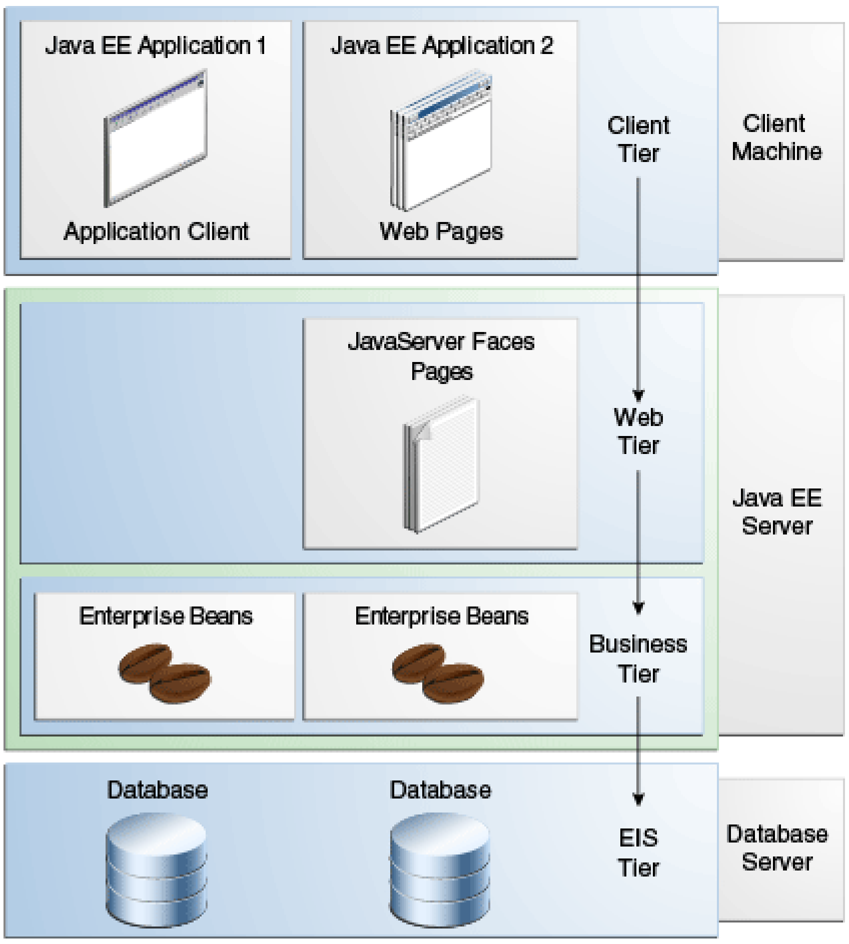


Figure 3-6-6: Multitiered Applications(Oracle 2014: Chapter 1.3)

The following class diagram illustrates the interfaces and implementation in the tiers of web and business. Those example classes are the Action, Service, DAO and Entity of Question and its super classes.

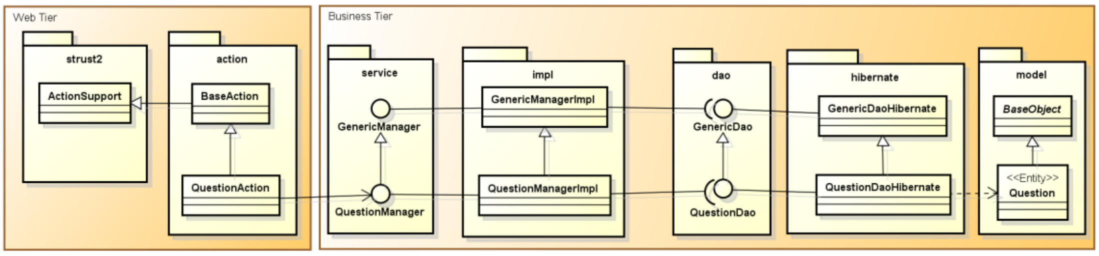


Figure 3-6-7: the example of implmentating Question in web and business tires

The reason of defining Entity classes is for mapping those Entities with Java syntax to tables in a relational database.

The reason of defining and implementing DAO interfaces is for manipulating the data stored in a relational database.

There are two reasons of defining and implementing the Service beans. The first is to bridge the DAO beans and Actions. The second is to decouple the web tier from business tier. The implementation of manager classes are supposed include the business logic for the application.

The Actions are defined for handling the requests from browsers.

The following items elaborate the details about the classes and relations between AppFuse and question module.

- The entity of question represents the data structure in a database. It is an entity class extending BaseOjbect.
- GenericDao and GnericDaoHibernate. They are interface and implementation class of Data Access Object (DAO) from AppFuse.
- QuestionDao and QuestionDaoHibernate. They, extending GenericDao and GnericDaoHibernate respectively, are interface and implementation classes of DAO for manipulating the data of question.
- GenericManager and GnericManagerImpl. They are interface and implementation class of service from AppFuse.
- QuestionManager and QuestionManagerImpl. They, extending GenericManager and GnericManagerImpl respectively, are interface and implementation classes of service for handling business computing of question.
- The class of ActionSupport. It is the default implementation of Action interface by Struts2.
- The class of BaseAction. It extends ActionSupport by AppFuse for providing some convenience methods.
- The class of QuestionAction. It extends BaseAction for responding the requests about manipulating question objects form client tier.

States of Template

As the information users inputting into filled forms are based on the structure of templates, for keeping the consistency of structure of various filled forms which are based on the same template, templates are not allowed to be changed after the templates are ready for users filling out. For simplifying the design and implantation, template is designed without the feature of deleting. In this way the implementation can avoid the complexity of cascade deleting operation on its questions and answers when a template is deleted. The following state diagram illustrates how the states of template changes.

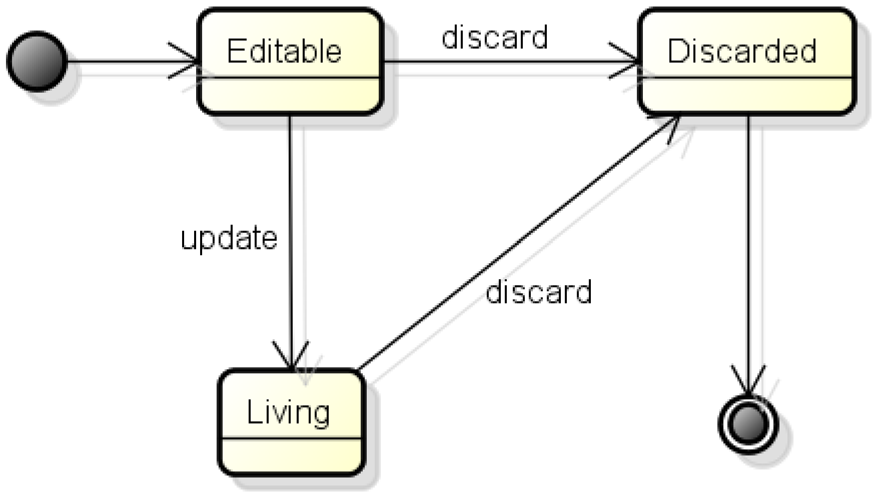
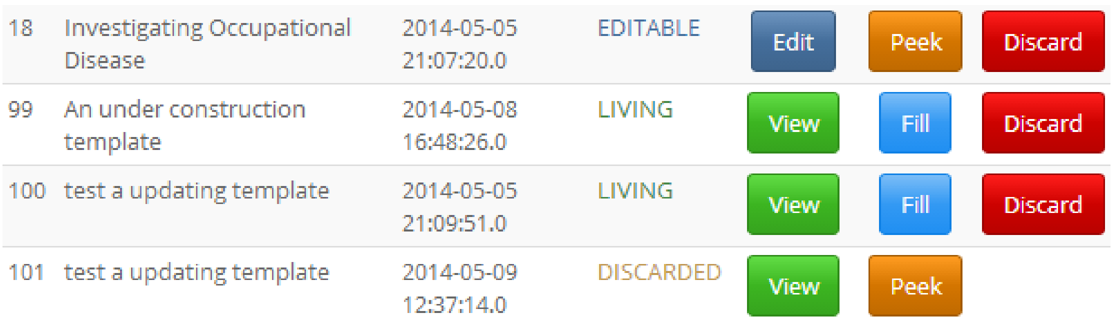


Figure 3-6-8: the state diagram of template

- The state of Editable is the default value when a template is created.
- When a template is 'Editable', user has three operations on the template. They are 'Edit', 'Peek' and 'Discard'.
    - The operation of 'Edit', for updating the template, may change the state of it to 'Living' or 'Discarded'.
    - The operation of 'Peek' is for having a look of the appearance of the template without changing its state.
    - The operation of 'Discard' changes its state to 'Discarded'.
- When a template is 'Living', user has three operations on the template. They are 'View', 'Fill' and 'Discard'.

  - The operation of ‘View’, for viewing the contents of template, does not change the state of it.
  - The operation of ‘Fill’ creates a new filled form.
  - The operation of ‘Discard’ changes its state to ‘Discarded’.
- The templates with the state of Editable or Living may change to Discarded by clicking the ‘Discard’ button on the template list page.
- The template with the state of ‘Discarded’ has two operations. They are ‘View’ and ‘Peek’ which are the same as above.

The following screenshot illustrates the operations of different template’s states on the list page.



18	Investigating Occupational Disease	2014-05-05 21:07:20.0	EDITABLE	Edit	Peek	Discard
99	An under construction template	2014-05-08 16:48:26.0	LIVING	View	Fill	Discard
100	test a updating template	2014-05-05 21:09:51.0	LIVING	View	Fill	Discard
101	test a updating template	2014-05-09 12:37:14.0	DISCARDED	View	Peek	

Figure 3-6-9: the operations of different template’s states on the list page

States of Question and Answer

The states of question and answer are changed following the state of template which they belong to. But the difference between template and question/answer is that question and answer have the feature of deleting. Thus the relations between a question and its answers are required to be removed, when the question is deleted. The implementation of it represents within the method of delete() in the file of dynamicform\web\src\main\java\uk\ac\le\student\zl91\webapp\action\QuestionAction.java. (From line 76 to 97)

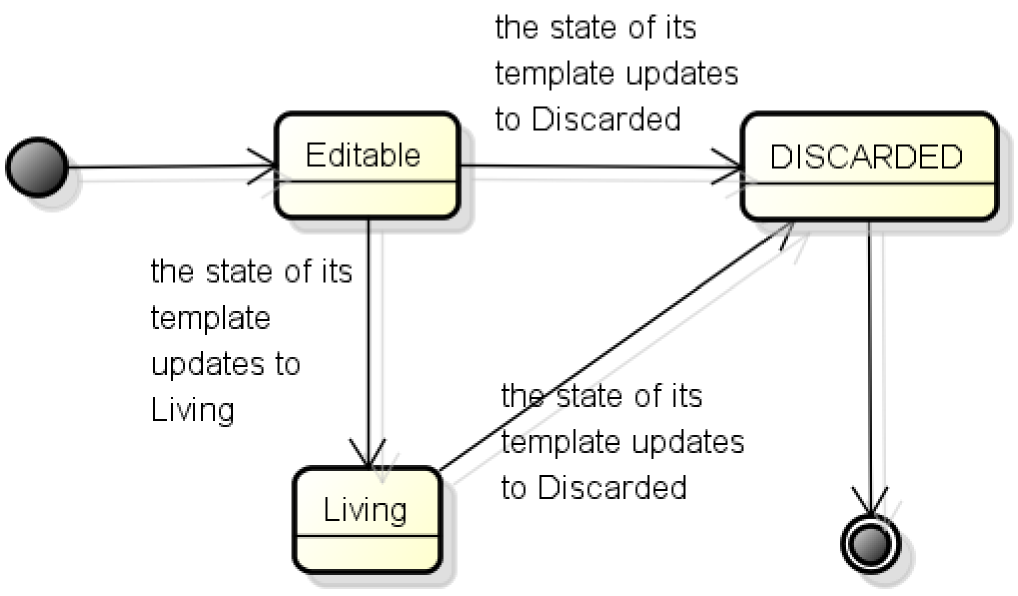
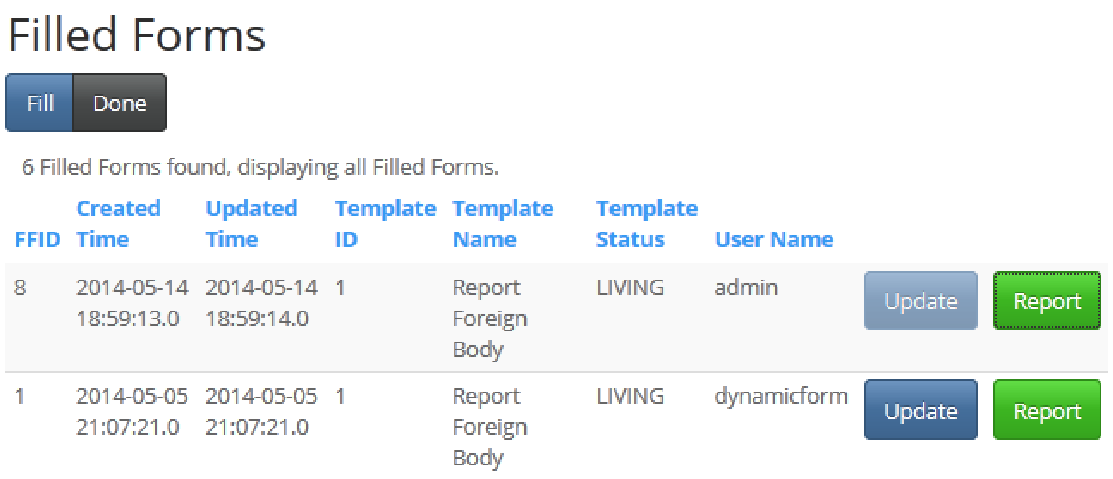

Editable
DISCARDED
Living
the state of its template updates to Discarded
the state of its template updates to Living
the state of its template updates to Discarded

Figure 3-6-10: the state diagram of question and answer

Users are only allowed to update their own filled form

In this application, any templates can be filled out by any users with any times. However only the owner of the filled form can update it, and other users have the privilege of viewing the filled form as a report. For example, the following screenshot illustrates that the user of dynamicform cannot update the filled form (FFID: 8) which is created by admin.



Filled Forms

Fill Done

6 Filled Forms found, displaying all Filled Forms.

FFID	Created Time	Updated Time	Template ID	Template Name	Template Status	User Name		
8	2014-05-14 18:59:13.0	2014-05-14 18:59:14.0	1	Report Foreign Body	LIVING	admin	Update	Report
1	2014-05-05 21:07:21.0	2014-05-05 21:07:21.0	1	Report Foreign Body	LIVING	dynamicform	Update	Report

Figure 3-6-11: Users are only allowed to update their own filled form

3.7 Implementation

As mentioned above, the implementation composites many technics and concepts. The following is a list for illustrating them respectively.

Collecting the inputs from a dynamic page

The one of the most complex implementation in the application is how to collect the inputs from a dynamic page and transform them to a list of object of Response entity. The implementation is the method of analyseResponse() in the file of dynamicform\web\src\main\java\uk\ac\le\student\zl91\webapp\action\FilledForm Action.java

At first, this paragraph explains how a web-based application collects the inputs usually. As a web-based application, the information transforming between browsers and servers is based on Hypertext Transfer Protocol (HTTP). The inputs transformed from browsers to servers are organised as pairs of key-value structure. In most of web-based applications, the elements of forms in the web pages are predefined and not change. For an instance, the login page has a form including the elements of username, password and a button of submit. After a user inputs ‘dynamicform’ as username, ‘df’ as password, and submit the form by clicking the submit button, the inputs are send to the server end as ‘username=dynamicform’ and ‘password=df’. The application can get the values of ‘dynamicform’ and ‘df’ according to the keys of ‘username’ and ‘password’ respectively. Under this circumstance, the keys of ‘username’ and ‘password’ are defined when programming, thus it is easy for getting the values.

Secondly, in this application templates are built up by users, therefore the filled forms contain dynamic elements. The number and name of elements are uncertain when programming, thus it is impossible to program the application collecting those inputs/values according to some curtain keys. For solving this problem, the application has to generate and analyse the keys according to a curtain rule. This rule is a mapping between a String and an identity of the element on the dynamic form. For identifying an element in form, three information are required the type of the element, the ids of question and answer. For an instance, a text field element in a dynamic form called ‘Site Name’ has the identity string of ‘TEXT|Q_21-A_21’. The ‘TEXT’ represents the type of the element is a text field. The sub-String of ‘A_21’ represents the Answer ID is 21. The sub-String of ‘Q_21’ represents the Answer is belong to the Question whose ID is 21.

In addition, for some questions with single or multi options, the value is required analysis as well. An example for a question with Radio buttons, the key is ‘RADIO|Q_902-GROUP’ and the value is ‘RADIO|Q_902-A_902’. Although the question with checkboxes allow user to select multi-answer, the way of analysis is similar as the way of Radio question. . There is an example for a checkbox question with two checkboxes selected. The one of key-value pair has the key of ‘CHECKBOX|Q_905-A_908’ and the value of ‘CHECKBOX|Q_905-A_908’, the other of key-value pair has the key of ‘CHECKBOX|Q_905-A_909’ and the value of ‘CHECKBOX|Q_905-A_909’.

Finally, on the dynamic form there is a special element called file which is utilised to upload a file. As there may be several file uploading elements in a dynamic form, the key of them share a same String as ‘FQMapping’, but the values are identified. For example the value of a file element is ‘Q_907-A_913|initProject.txt*’ which uploads a file named as ‘initProject.txt’.

The following is the method of analyseResponse().

```
private List<Response> analyseResponse(){
    List<Response> responseList = new ArrayList<Response>();
    Map<String,String[]> map = this.getRequest().getParameterMap();
    for (Iterator<String> it = map.keySet().iterator(); it.hasNext(); ) {
        String key = it.next();
        String value = "";
        if (map.get(key).length==1){
            value = map.get(key)[0];
        }
        else if(map.get(key).length>1){
            log.error("Impossible value:"+Arrays.toString(map.get(key)));
        }
        log.debug("Answer key:"+key+" value:"+value);
        //FQMapping , AnswerType
        if (key.equals("FQMapping")){
            responseList.addAll(
                    this.buildResponse(
                            this.matchUploadedFiles(this.uploads,
                                    this.uploadFileNames,
                                    this.uploadContentTypes,
                                    value)));
        }
        else {
            String type =getTypeFromResponseKey(key);
            if(AnswerType.RADIO.toString().equals(type)){
                String cid = getCombinedIdFromResponse(value);
                responseList.add(
                    this.buildResponse(cid,"checked")
                );
            }
            else if(AnswerType.CHECKBOX.toString().equals(type)){
                String cid = getCombinedIdFromResponse(key);
                responseList.add(
                    this.buildResponse(cid,"checked")
                );
            }
            else if(AnswerType.TEXTAREA.toString().equals(type)){
                String cid = getCombinedIdFromResponse(key);
                responseList.add(
                    this.buildResponse(cid,value)
                );
            }
            else if(AnswerType.TEXT.toString().equals(type)){
                String cid = getCombinedIdFromResponse(key);
                responseList.add(
                    this.buildResponse(cid,value)
                );
            }
            else if(AnswerType.FILE.toString().equals(type)){
                //do nothing
            }else{
                log.debug("non-dynamic key:"+key);
            }
        }
    }
    return responseList;
}
```

Code segment 3-7-1: the method of analyseResponse() in the class of FilledFormAction.java

- The method of buildResponse() is an overload method with two different group of parameters.
    - The one with parameters of ‘List<UploadedFile>’ reads the uploading files from the disk on server side and transform each UploadedFile object to the object of Response.
    - The other with the parameters of ‘String key, String value’ generates the new Response object according to the key-value pair.
- The method of matchUploadedFiles() analyses and transforms the uploaded files into a list of UploadedFile.
- The method of getTypeFromResponseKey() get the pre-defined types of AnswerType within the key.
- The method of getCombinedIdFromResponse() get the string of combined ids of question and answer. For example it will return the string of combined ids as ‘Q_5-A_11’ from the input string of ‘CHECKBOX|Q_5-A_11’

Generating the dynamic page with a user's inputs

Another of the most complex implementation in the application is how to generate the dynamic page with the inputs which user saved at the last update. The implementation has two parts. One is the method of initDynamicFormParameters() in the file of dynamicform\web\src\main\java\uk\ac\le\student\zl91\webapp\action\FilledFormAction.java. Another is the file of \dynamicform\web\src\main\webapp\WEB-INF\pages\filledFormView.jsp

On one hand, the filled form consists of a number of questions. A question may have several sub-questions or answers. Each answer may have some inputs (called Response in the software) which a user filled at the last update. Therefore for building a dynamic form with the last updating content, the method of buildFilledFormView() iterates three levels of objects of Question, Answer and Response. In the lowest level of iterating, the program gets and then sets the transient field of content to Answer. The following illustrates the method of buildFilledFormView().

```java
private List<Question> buildFilledFormView(FilledForm filledForm){

    //pass the content from responses to answers, By
    //setting value of FilledForm->Response.content
    //to FilledForm->Template->Question->Answer.content
    Set<Question> tquestionSet= filledForm.getTemplate().getQuestionSet();
    Set<Response> responseSet = filledForm.getResponseSet();
    for(Iterator<Question> itq = tquestionSet.iterator(); itq.hasNext(); ) {
        Question tq = itq.next();
        Set<Answer> tqanswerSet = tq.getAnswerSet();
        for(Iterator<Answer> ita = tqanswerSet.iterator(); ita.hasNext(); ) {
            Answer tqa = ita.next();
            for(Iterator<Response> itr = responseSet.iterator(); itr.hasNext(); ) {
                Response r = itr.next();
                //put the content and rid of a response to the answer
                if (r.getAnswer().getAid() == tqa.getAid()
                        && r.getQuestion().getQid() == tq.getQid()){
                    tqa.setContent(r.getContent());
                    tqa.setRid(r.getRid());
                }
            }
        }
    }
    List<Question> filledFormView = this.buildTemplateView(filledForm.getTemplate());
    return filledFormView;
}
```

Code segment 3-7-2: the method of buildFilledFormView() in the class of FilledFormAction.java

In the method of initDynamicFormParameters(), the program passes the question list, returning from method of buildFilledFormView() , to the JSP file of dynamicform\web\src\main\webapp\WEB-INF\pages\filledFormView.jsp by the code of ‘this.getRequest().setAttribute("filledFormView",filledFormView);’

On the other hand, three methods are defined in the filledFormView.jsp for generating the dynamic content of form. They are

- outputQuestion() generates all the HTML elements according to an object of question which may including several answers or sub-questions.
- outputAnswerInLine() generates all the HTML elements according to an object of answer with the answer refers to
- outputAnswer()generates all the HTML elements according to the various type of a single answer.

The object of question list is passed as value to the key ‘filledFormView’ from the action class mentioned above to the page of filledFormView.jsp. Then each question in the list is transformed by method of outputQuestion() and outputs as HTML elements. The following codes is the implementation.

```
<%!
    private String voidNull(String s) {
        return (s == null || "null".equalsIgnoreCase(s)) ? "" : s;
    }
        /**
         * out put an answer on page.
         * The id of answer should be Q_[qid]_A_[aid] for example Q_2_A_4
         * @param answer
         * @return
         */
    private String outputAnswer(Answer answer) {
        String as = new String();
        String type = answer.getType().toString();
        String qid_s= "Q_" + answer.getQuestion().getQid();
        String qid_s_group= answer.getType()+"|"+qid_s+"-GROUP";
        String aid_s= "A_" + answer.getAid();
        String id = answer.getType()+"|"+qid_s +"-" + aid_s;


        if (AnswerType.RADIO == answer.getType()) {

            as = as
                    + "<div class=" + type + ">\n"
//                    + "  <label>\n"
                    + "    <input type=\"" + type + "\" name=\""+qid_s_group+"\""
                    +"  value=\""+id+"\" "
                    + this.voidNull(answer.getContent())+ ">\n"
//                    +answer.getAid() + ":"
                    + answer.getTitle()
//                    + "</label>\n"
                    + "</div>"
            ;

        } else if (AnswerType.TEXT == answer.getType()) {
            String lable =
                    (answer.getTitle() != null && answer.getTitle().trim().length() != 0) ?
```

```
                            "<div class=\"col-sm-4 text-right control-label\">" +
answer.getTitle() + "</div>\n" :
                            "";
            String hintContent =
(answer.getHint()==null||answer.getHint().trim().length()==0) ? "" : answer.getHint();
            String hint =  " placeholder=\""+hintContent+"\"";
            String value = (answer.getContent()==null||
            "null".equalsIgnoreCase(answer.getContent())) ? "":answer.getContent();
            as = as
                    + "<div class=\"" + type + "\">\n"
                    + lable
                    + "  <input type=\"" + type + "\" " +" name=\""+id+"\""
                    + " value=\""+value+"\""
                    + " class=\"form-control\" "
                    + hint
                    + ">\n"
                    + "  </input>\n"
                    + "</div>\n"
            ;
        } else if (AnswerType.CHECKBOX == answer.getType()) {
            as = as
                    + "<div class=\"" + type + "\">\n"
                    + "  <div>\n"
                    + "    <input type=\"" + type  + "\" name=\""+id+"\" value=\""+id+"\""
                    +  this.voidNull(answer.getContent())+"/>\n"
                    + answer.getTitle()
                    + "  </div>\n"
                    + "</div>\n"
            ;
        } else if (AnswerType.TEXTAREA == answer.getType()) {
            String hintContent =
(answer.getHint()==null||answer.getHint().trim().length()==0) ? "" : answer.getHint();
            as = as
                    +"  <"+ type
                    + " placeholder=\""+hintContent+"\""
                    + " name=\""+id+"\" class=\"form-control\" rows=\"3\">"
                    +  this.voidNull(answer.getContent())
                    +"</textarea>\n"
            ;
        } else if (AnswerType.FILE == answer.getType()) {
            String downloadButton=
                    (this.voidNull(answer.getContent())=="")?"":
                    "  <div class=\"col-sm-4\">\n"
                    +"     <a class=\"btn btn-success\"
href=\"/download?rid="+answer.getRid()+"\" role=\"button\">Download "+
this.voidNull(answer.getContent())+"</a>"
                    +"  </div>\n";
            as = as
                    +"<div class=\"row\">\n"
                    +"  <div class=\"col-sm-4\">\n"
                    +"    <input class=\"btn btn-info\" type=\"" + type+ "\" name=\"upload\""
                    +" id=\""+qid_s +"-" + aid_s+"\"></input>\n"
                    +"  </div>\n"
                    + downloadButton
                    +"</div>\n"
            ;
        } else if (AnswerType.LABEL == answer.getType()) {
            as = as
//            control-label
                    + "<div class=\"text-right\">"
                    + this.voidNull(answer.getTitle())
                    + "</div>"
            ;
        }

        return as;
    }
    private String outputAnswerInLine(Answer answer) {
        String as = new String();
        as = as +"<div class=\"row form-group\">";
//        as = as + outputAnswer(answer);
        Answer tempa = answer;
        while (tempa!= null) {
            as = as +"<div class=\"col-sm-"+tempa.getWidth()+"\">";
            as = as + outputAnswer(tempa);
            as = as +"</div>";
            tempa = tempa.getReferredAnswer();
        }
        as = as +"</div>";
        return as;
    }

    private String outputQuestion(Question question) {

        boolean isBlankQuestion = false;
        String qs = new String();

        String title  = question.getTitle();
        //the question without title means for grouping the radio answers
        if ((title == null || "null".equalsIgnoreCase(title)
        || title.trim().length()==0) ) {
            isBlankQuestion = true;
        }

        if (!isBlankQuestion){
            qs = qs + "<div class=\"form-group\">"
                    + "<label class=\"control-label\">"+question.getTitle() + "</label>"
            ;
        }

//          The Answers belong to the current question
        for (Iterator<Answer> it = question.getAnswerSet().iterator(); it.hasNext(); ) {
            Answer answer = it.next();
            qs = qs + outputAnswerInLine(answer);
        }

        if (!isBlankQuestion) {
            qs = qs + "</div>";
        }

        if (question.getQuestionSet().size()>0){
            System.out.println("ParentQid:"+question.getQid());
            List<Question> questionList = new ArrayList<Question>(question.getQuestionSet());
            Collections.sort(questionList);
            qs = qs + "<div class=\"row\">"
                    + "   <div class=\"col-sm-1\"></div>"
                    + "   <div class=\"col-sm-11\">";
            for (int i = 0; i < questionList.size(); i++) {
                System.out.println("ChildQid:"+questionList.get(i).getQid());

                qs = qs + outputQuestion(questionList.get(i));
            }
            qs = qs+ "  </div>"
                    + "</div>";

        }

        return qs;
    }
%>


<%--receive the question list for building the dynamic form--%>
<%List<Question> filledFormView = (List<Question>) request.getAttribute("filledFormView");%>
<%  //generating the content of form question by question
      Collections.sort(filledFormView);
      for (int i = 0; i < filledFormView.size(); i++) {
          Question question = filledFormView.get(i);
          out.println(outputQuestion(question));
      }
%>
```

Code segment 3-7-3: Some Java code in the page of filledFormView.jsp

In addition, A JavaScript method of fqMapping() is defined at the bottom of this JSP file. The function of the method is for collecting the names of uploading files from the elements named as 'upload' and set the elements' IDs with names into a hidden field called 'FQMapping' which mentioned above. Thanks to this method, the action can receive the uploading names. For an instance, the application can get the uploading name with question and answer ids with a String of 'Q_907-A_913|initProject.txt*' from the key of 'FQMapping'.

Maven

Maven as a tool is introduced into the process of development. For initialising the development environment of the application, the archetype plugin of Maven is utilised through the command of Maven as following. It is generated on the web page of 'AppFuse QuickStart' (Raible 2014).

```
mvn archetype:generate -B -DarchetypeGroupId=org.appfuse.archetypes -
DarchetypeArtifactId=appfuse-modular-struts-archetype -DarchetypeVersion=3.0.0 -
DgroupId=uk.ac.le.student.zl91 -DartifactId=dynamicform -
DarchetypeRepository=http://oss.sonatype.org/content/repositories/appfuse
```

Command segment 3-7-4: the command of Maven for initialising the development environment

Several parameters in this command are critical. They are DarchetypeArtifactId, DarchetypeVersion, DgroupId and DartifactId.

- DarchetypeArtifactId defines two optional items. The first is whether the application is multi-module. The 'modular' means it has two modules. The one is 'core'. The other is 'web'. If it is a single-module, the parameter should be 'basic'. The second is about which web framework are introduced in the application. The optional items are 'core', 'jsf', 'spring', 'struts', 'tapestry', 'wicket' and 'ws'.
- DarchetypeVersion has three optional version of AppFuse. There are versions of '2.2.1', '3.0.0', '3.0.1-SNAPSHOT'
- DgroupId is the name space for the application following the specification of naming java package.
- DartifactId is the name of the application following the specification of naming java variables.

Maven facilitates all the tasks during the development of application such as cleaning, building, debugging, testing and running. The following are some examples used in the process of developing the application.

- Run the task of cleaning and building for core module (at the path of dynamic\core) as the following:

```
mvn clean install
```

Command segment 3-7-5: the command of Maven for cleaning and building core module

- Run the task of building for web module (at the path of dynamic\web) and avoiding testing as the following:

```
mvn clean install -DskipTests
```

Command segment 3-7-6: the command of Maven for building core module and avoiding testing

- Debugging the application (at the path of dynamic\web) and avoiding testing as the following:

```
mvnDebug jetty:run -DskipTests
```

Command segment 3-7-7: the command of Maven for debugging the application and avoiding testing

- Run a specific test class of the application (at the path of dynamic\core) as the following:

```
mvn test -Dtest=QuestionDaoTest
```

Command segment 3-7-8: the command of Maven for running a specific test class

JPA and Entity

There are seven entity classes in the application. Two of them, User and Role, are from Appfuse. The rest of five entity classes, Question, Answer, Template, FilledForm, and Response are created by the author in the folder of dynamicform\core\src\main\java\uk\ac\le\student\zl91\model. All of those entity classes are created with two JPA annotations. The one is '@Entity', the other is '@Table'

- '@Entity' defines the current class is an entity for persistent layer.
- '@Table' defines the name of table mapped into a database.

- The annotation of '@Id' is for mapping the current variable as a primary key into a table.
- The annotation of '@Column' is for mapping the current variable as a column into a table.
- The annotation of '@OneToOne', '@OneToMany' and '@ManyToOne' are for mapping the relation between two entities.
- The annotation of '@JoinTable' is for introducing a new join table for representing the relation between two entities.
- The annotation of '@Transient' is for temporarily keep a value with the entity object which cannot be persistent.
- The annotation of '@Override' is not a JAP annotation. It is helpful to the compiler knowing the following method is an override method.

Some Entity classes are defined with Java Persistence query language (JPQL) named queries. 'JPQL named queries can be defined in the entity class using a Java programming language annotation or in the application's deployment descriptor.'(Jendrock, Cervera-Navarro et al. 2014: 37-19) The following is an example of two named queries defined in entity of Answer.

```
@Entity
@Table(name = "answer")
@NamedQueries(
        {
                @NamedQuery(name = "Answer.findByQuestion", query = "SELECT answer FROM Answer
answer WHERE answer.question.qid = :qid"),
                @NamedQuery(name = "Answer.findAvailableAnswer",
                        query = "select answer      from Answer answer "
                                + "                     left join answer.question question "
                                + "                     left join question.template template "
                                + "                     where   (template.status is null or
template.status = 'EDITABLE')"
                )
        }
)
public class Answer extends BaseObject implements Comparable<Answer>{
    //the primary key of the table is aid
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long aid;

//    the foreign key from question.qid
    @ManyToOne//(optional = false)//filledForm is the entity object in the Response class
    @JoinColumn(name = "qid")// unique = false(default)
    private Question question;

    public Question getQuestion() {
        return question;
    }

    public void setQuestion(Question question) {
        this.question = question;
    }

    //there are 6 types of an answer: RADIO, CHECKBOX, TEXT, TEXTAREA, FILE, LABEL
    //the LABEL is for the needs of layout.
    //the default type is RADIO.
    @Column(name = "type",
            columnDefinition="varchar(15) NOT NULL DEFAULT 'RADIO'")
    @Enumerated(EnumType.STRING)
    private AnswerType type=AnswerType.RADIO;

    //the title of the answer which show before/after the element of the HTML.
    //it works for all of the types expect FILE.
    @Column(length = 1000)
    private String title;

    //the hint shows as text in the element of the HTML.
    //it works on the types of TEXT and TEXTAREA.
    @Column(length = 1000)
    private String hint;

    //the number of order on the answer. it defines the order of answers showing in a question.
    @Column(name = "order_number",columnDefinition = "int NOT NULL DEFAULT 0")
    private Integer orderNumber=0;

    //when the answer is created, which is a good identity for distingrish the objects of
answer
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "create_time",columnDefinition = "timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP")
    private Date createTime = new Date(System.currentTimeMillis());

    //the default value of a element of the HTML.
    //it works on the types of RADIO, CHECKBOX, TEXT and TEXTAREA.
    @Column(name = "default_value", length = 1000)
    private String defaultValue;

    //for the layout each element of HTML on a page, the width should between 1 and 12
    @Column(name = "width",columnDefinition = "int NOT NULL DEFAULT 12")
    private Integer width=12;

    //one answer may have many responses
    @OneToMany(mappedBy = "answer")//answer is the entity object in the Response class
    private Set<Response> responseSet;

    //one answer may control a question
    //the foreign key from question.qid
    @OneToOne
    @JoinColumn(name = "referred_qid", nullable = true)
    private Question referredQuestion;

    public Question getReferredQuestion() {
        return referredQuestion;
    }

    public void setReferredQuestion(Question referredQuestion) {
        this.referredQuestion = referredQuestion;
    }


    @OneToOne
    @JoinColumn(name = "referred_aid")
    private Answer referredAnswer;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Answer answer = (Answer) o;

        if (createTime != null ? !createTime.equals(answer.createTime) : answer.createTime !=
null) return false;
        if (defaultValue != null ? !defaultValue.equals(answer.defaultValue) :
answer.defaultValue != null)
```

```
            return false;
        if (hint != null ? !hint.equals(answer.hint) : answer.hint != null) return false;
        if (orderNumber != null ? !orderNumber.equals(answer.orderNumber) :
answer.orderNumber != null) return false;
        if (width != null ? !width.equals(answer.width) : answer.width != null) return false;
        if (title != null ? !title.equals(answer.title) : answer.title != null) return false;
        if (type != answer.type) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = type != null ? type.hashCode() : 0;
        result = 31 * result + (title != null ? title.hashCode() : 0);
        result = 31 * result + (hint != null ? hint.hashCode() : 0);
        result = 31 * result + (orderNumber != null ? orderNumber.hashCode() : 0);
        result = 31 * result + (width != null ? width.hashCode() : 0);
        result = 31 * result + (createTime != null ? createTime.hashCode() : 0);
        result = 31 * result + (defaultValue != null ? defaultValue.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "Answer{" +
                "aid=" + aid +
                ", type=" + type +
                ", title='" + title + '\'' +
                ", hint='" + hint + '\'' +
                ", orderNumber=" + orderNumber +
                ", createTime=" + createTime +
                ", defaultValue='" + defaultValue + '\'' +
                ", width=" + width +
                '}';
    }


    @Override
    public int compareTo(Answer a) {
        return this.orderNumber-a.orderNumber;
    }

    //the content is for temporarily keep the value of content from response entity.
    @Transient
    private String content;

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    //the content is for temporarily keep the response id of response entity.
    @Transient
    private long rid;//for downloading file

    public long getRid() {
        return rid;
    }

    public void setRid(long rid) {
        this.rid = rid;
    }
}
```

Code segment 3-7-9: the entity class of Answer

The annotation of ‘NameQueries’ includes an array of annotations of ‘NamedQuery’. Each of ‘NamedQuery’ consists of name and query. The value of name represents the name of named query. The query is a JPQL.

For an entity class with a generated primary key, implementing the methods of ‘equals’ and ‘hashCode’ is required (Jendrock, Cervera-Navarro et al. 2014: 37-6) . The purpose of implementing the two methods is for identifying the duplication of entity objects within a collection such as Set. In addition, the class of ‘BaseObject’ from AppFuse is an abstract class with three abstract methods. They are ‘equals’, ‘hashCode’ and ‘toString’. Therefore any sub class, extending ‘BaseObject’ class, has to implement them.

The method of compareTo() is required to override by the interface of Comparable. This method return an integer value which determined the order between two objects of the current one and the one passed as a parameter. Therefore it is utilised to sort the objects within an ordered collection such as List. The implementation of this method defines the order of answers belonged to a question according to the the property of ‘orderNumber’ increasingly.

DAO

The purpose of DAO is for manipulating single entities, which includes the operations of creating, updating, deleting and requiring on entities. There are two ways to define a DAO object/bean in the application. One is registering a DAO bean with the class of ‘GenericDaoHibernate’. The other is creating and implementing a specific DAO interface, and then registering the specific DAO bean to the application. Those two ways are applied into the application for different purposes. Those beans are defined in the file of ‘applicationContext.xml’. The details are illustrated below.

- Registering a DAO bean with ‘GenericDaoHibernate’. It provides a basic method of manipulating data with a few operations of create, read, update and delete (CRUD). The examples in the dynamic form application are beans of ‘templateDao’, ‘filledFormDao’ and ‘responseDao’.
- Registering a specific DAO bean. It does not only provide the operations of CRUD, but also allow developers defined customised methods to manipulate

data. Actuarially, the latter is the reason that the specific DAO beans are introduced into the application. For examples 'answerDao' and 'questionDao' have a method named 'getList' which return all the entity objects of answer or question. The classes of 'AnswerDaoHibernate' and 'QuestionDaoHibernate' are the implementations of the DAO beans, as both of them have the annotation of '@Repository' which relates the name of bean to the current class. For example @Repository("answerDao") in the class of AnswerDaoHibernate, which relates the bean of 'answerDao' to the implementation of 'AnswerDaoHibernate'.

Service

The service is introduced for implementing the business computing within a system. As the same as DAO, there are two ways to define Business Facade classes in the application. One is registering a service bean with the class of 'GenericManagerImpl'. The other is creating and implementing a specific Service interface, and then registering the specific Service bean to the application. Those two ways are applied into the application for different purposes. Those beans are defined in the file of 'applicationContext.xml'. The details are illustrated below.

- Registering a Service bean with 'GenericManagerImpl'. It simply bridges the DAO beans and Actions by adapting the interface of GenericDao. The examples in the dynamic form application are beans of 'filledFormManager' and 'responseManager'.
- Registering a specific Service bean. It does not only provide a bridge between DAO beans and Actions, but also allow developers defined customised methods to manipulate data. Actuarially, the latter is the reason that the specific Service beans are introduced into the application. For examples 'templateManager' have a method named 'findAvailableTemplate' which return all the templates with the state of EDITABLE. The classes of 'AnswerManagerImpl', 'QuestionManagerImpl' and 'TemplateManagerImpl' are the implementations of the Service beans, as all of them have the annotation of '@Service' which relates the name of Service bean to the

current class. For example @Service("templateManager") in the file of TemplateManagerImpl.java

Action

There are four actions introduced for implementing the application in the folder of dynamicform\web\src\main\java\uk\ac\le\student\zl91\webapp\action. They are FilledFormAction, AnswerAction, QuestionAction and TemplateAction. All of them extend BaseAction which is from AppFuse for providing some common functions and current user information. All the actions register to the Struts2 configure file of struts.xml.

- FilledFormAction is for handling the requests of
    - Creating, saving and deleting a single filled form object.
    - Listing a certain number of the objects of filled form.
    - Preparing the data shown on the update, report, fill and peak pages of filled form.
    - A method of download() is for user downloading the file which is uploaded at last updating.
    - This action is one of the most complex part in the application as the following reasons:
        - The content of form is dynamic, therefore organising the objects of the content is a challenge.
        - When representing a filled form for the purposes of updating or printing, it is not only showing the organised objects of the content, but also representing those inputs which user had committed early.
        - As the elements/markups on the form page are dynamic, they also have a dynamic name and type. It makes collecting the inputs on the dynamic page harder when a user submits the dynamic form.
        - An inner class of UploadedFile for containing the information about the uploaded/download file.
- AnswerAction is for handling the requests of

    - Creating, saving and deleting a single answer object.
    - Listing a certain number of the objects of answer.
    - Preparing the data shown on the edit/view page of answer.
- QuestionAction is for handling the requests of
    - Creating, saving and deleting a single question object.
    - Listing a certain number of the objects of question.
    - Preparing the data shown on the edit/view page of question.
- TemplateAction is for handling the requests of
    - Creating, saving and discarding a single template object.
    - Listing a certain number of the objects of template.
    - Preparing the data shown on the edit/view page of template.

JSP

The JSP files implemented in the application consist of several technics such as Bootstrap, JSTL, OGNL and JSP API 2.0. Nine JSP files are created and introduced for implementing the application in the folder of \dynamicform\web\src\main\webapp\WEB-INF\pages. The details are illustrated as following

- filledFormList.jsp is for the purpose of representing the list of filled forms.
- filledFormView.jsp is for the purpose of creating a new form to fill, peeking a specific template, and updating/deleting a specific filled form. This page is the most complex page in the application.
- filledFormReport.jsp, simplified edition of filledFormView.jsp, is for the purpose of representing a report page for the filled form.
- templateList.jsp is for representing the list of templates.
- templateForm.jsp is for the purpose of creating a new template or viewing/editing a specific template.
- questionList,jsp is for the purpose of representing the list of questions.
- questionForm.jsp is for the purpose of creating a new question or viewing /editing/deleting a specific question.
- answerList.jsp is for the purpose of representing the list of answers.

- answerForm.jsp is for the purpose of creating a new answers or viewing /editing/deleting a specific answers.

In addition, those JSP files are developed with the paradigm of responsive web pages which are supported by Bootstrap. It not only adapts to a range of width, but also supports the hidden irrelevant elements from printing. For example, on the report page (filledFormReport.jsp) the HTML elements of menu bar, foot and buttons of ‘Print’ & ‘Back’ are defined with the value ‘hidden-print’ with the attribute name of ‘class’.

Multi-language

AppFuse supports the future of multi-language which includes sixteen different languages such as English, French, Italian, Chinese, etc. the ‘dynamic form’ software inherits this feature. The name of properties files following the format ‘ApplicationResources_**.properties’ is for supporting a certain language. The properties file of ApplicationResources.properties in the folder of dynamicform\web\src\main\resources is the default language for the software. Since there is no requirement about supporting other languages in the software, only English is implemented as default language within ‘dynamic form’. The syntax of the properties file is a pair of key-value and the comments start with symbol ‘#’. The part of value allows developers to pass an arbitrary number of parameters to it. The number of parameters starts from zero with a specific format ‘{*number*}’. For example, the first parameter presents as ‘{0}’.

The fmt, one of JSTL tags, reads a certain properties file according to the default setting of the users’ browser and represents the value according to the key in the properties file. The following is an example of representing the text on the fill button on the page of filledFormList.jsp. <fmt:message key="filledFormList.button.fill"/>

3.8 Logging and Debugging

There are two ways for debugging during the process of developing the software. One is logging, the other is remote debugging.

Logging

For the purpose of tracking and debugging program, the logging is introduced into the software. The log object of org.apache.commons.logging.Log is defined in the classes of GenericDaoHibernate, QuestionManagerImpl and BaseAction which are super classes for the implementations of DAO, Service and Action classes respectively. Their sub-classes, therefore, inherit and use it directly. On one hand the log object is following the interface of commons-logging[18] from Apache, as the evidence is the Log class coming from the library of commons-logging:commons-logging:1.1. On the other hand the implementation is by Log4j[19] with the library of log4j:log4j:1.2.17.

In addition, the configuration file of log4j in the folder of dynamicform\web\src\main\resources is updated for enabling the log codes written in the classes for the application. For removed the information of thread, the output format is changed slightly. The default pattern is "%p [%t] %c{1}.%M(%L) | %m%n". This is the pattern after changed "%p %c{1}.%M(%L): %m%n".

- The '%c{1}' represents the name of class which the logging code is in.
- The '%M' represents the name of method which the logging code is in.
- The '%L' represents the number of line which the logging code is at.
- The '%m' represents the message of within the log.
- The '%n' represents a new line.

The following XML segment is added into the file for outputting the logs for dynamic form. <logger name="uk.ac.le.student.zl91"> <level value="DEBUG"/> </logger>

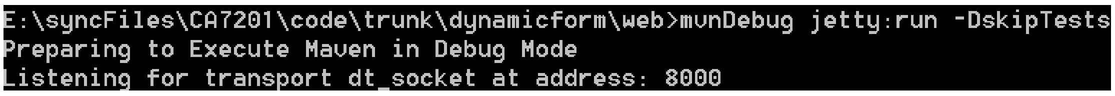
[18] APACHE, 2014-last update, Commons Logging. Available: http://commons.apache.org/proper/commons-logging/ [5/6/2014, 2014]

[19] APACHE, 2014-last update, Log4j. Available: https://logging.apache.org/log4j/1.2/ [5/6/2014, 2014]

Debugging

Debugging the application depends on two tools. One is the Surefire[20] Plugin of Maven. The other is a Java software development tool named IntelliJ IDEA[21]. There are three steps for building up the environment for debugging the software.

At first, lunching the application under the debug mode by enter the Maven command 'mvnDebug jetty:run -DskipTests'. The following screenshot represents the application running under the debug mode and listening at port 8000.



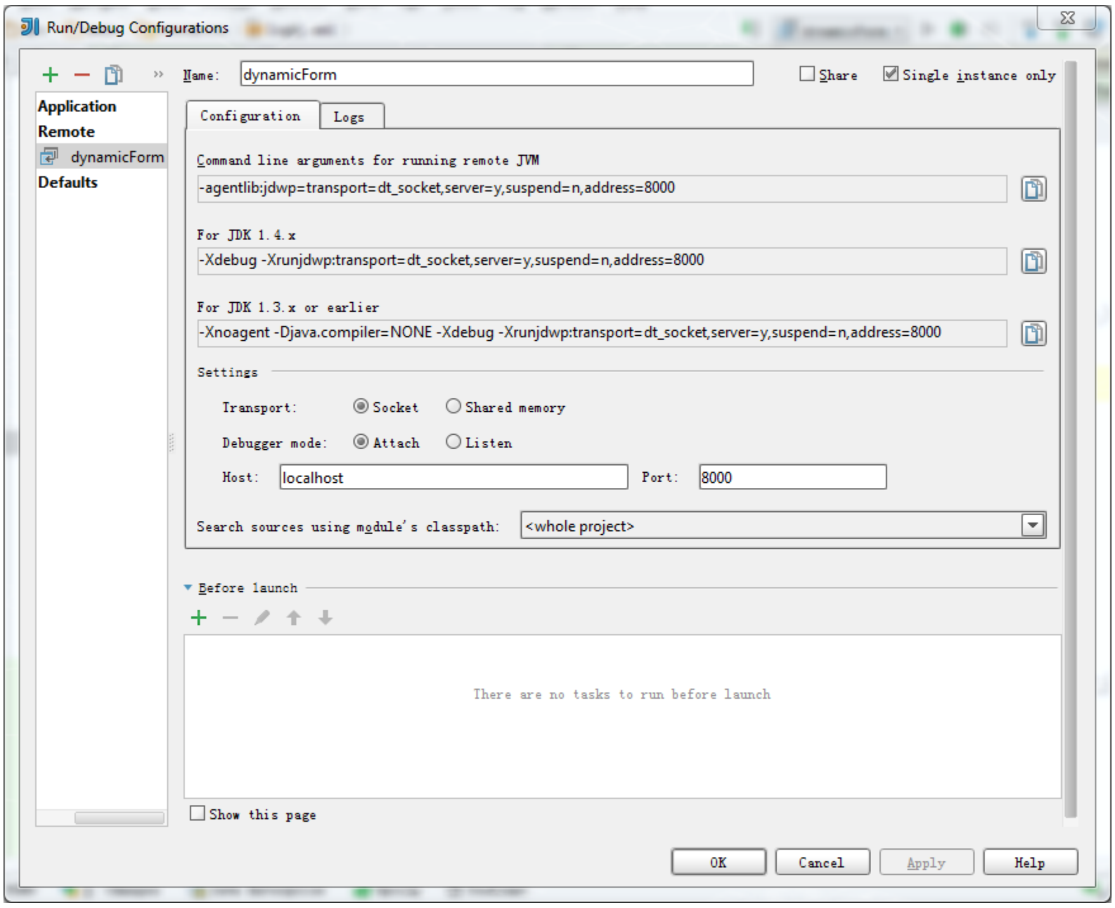
```
E:\syncFiles\CA7201\code\trunk\dynamicform\web>mvnDebug jetty:run -DskipTests
Preparing to Execute Maven in Debug Mode
Listening for transport dt_socket at address: 8000
```

Figure 3-8-1: the screenshot of application is under debugging mode

Secondly, create a new debug configuration in IntelliJ IDEA with the parameters shown as the following screenshot.

[20] APACHE, 2014-last update, Maven Surefire Plugin. Available: http://maven.apache.org/surefire/maven-surefire-plugin/index.html [5/6/2014, 2014]

[21] JETBRAINS, 2014-last update, IntelliJ IDEA. Available: http://www.jetbrains.com/idea/ [5/6/2014, 2014]



Name: dynamicForm
[ ] Share [✓ Single instance only]

Command line arguments for running remote JVM
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000

For JDK 1.4.x
-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000

For JDK 1.3.x or earlier
-Xnoagent -Djava.compiler=NONE -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000

Settings
Transport: [✓ Socket] Shared memory
Debugger mode: [✓ Attach] Listen
Host: localhost Port: 8000
Search sources using module's classpath: <whole project>

[ ] Show this page

Figure 3-8-2: the screenshot of debug configuration

Finally, in IntelliJ IDEA start the debugging by clicking the debugging button shown as bellow.

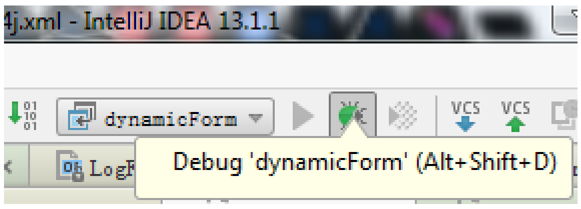
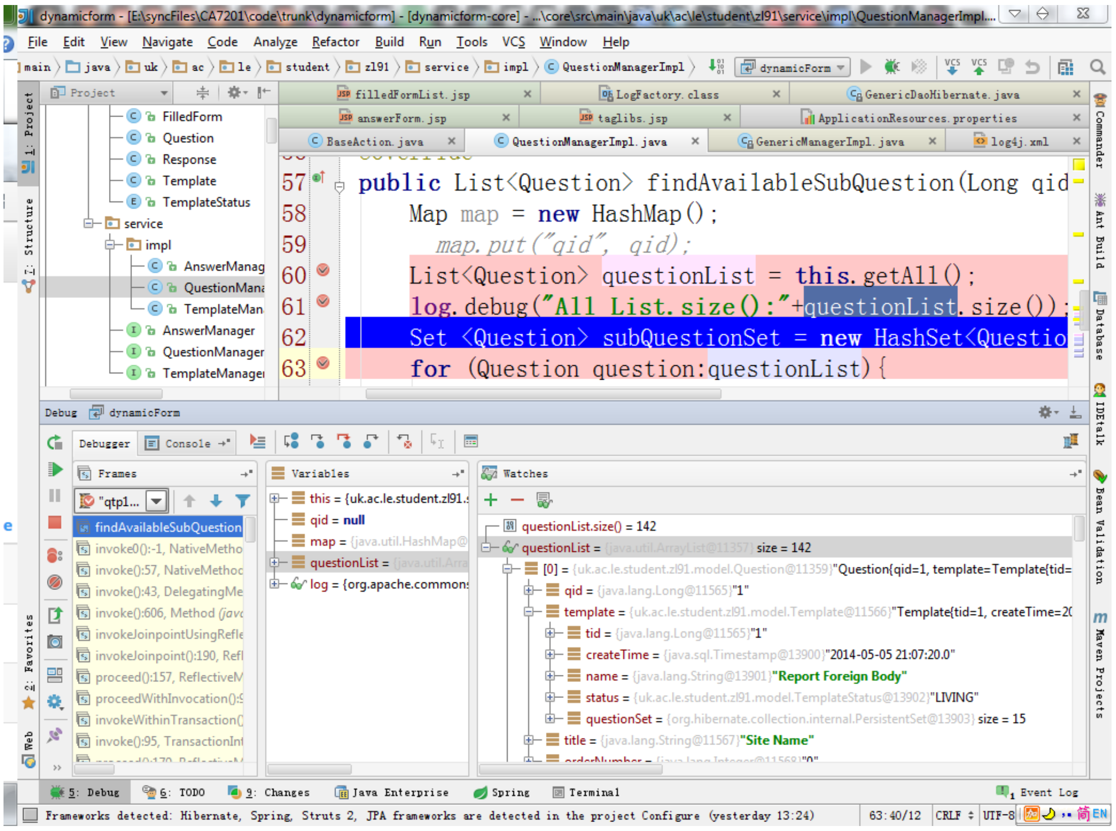

4j.xml - IntelliJ IDEA 13.1.1
dynamicForm
VCS
VCS
LogF
Debug 'dynamicForm' (Alt+Shift+D)

Figure 3-8-3: the screenshot of debugging button in IntelliJ IDEA

For debugging and tracking some variables or methods within the software, it is necessary to add some break-points in the codes. The following screenshot represents three break-points are set in the method of findAvailableSubQuestion in the class of QuestionManagerImpl. After the program stop at the break-point, IntelliJ IDEA allows developers to run the program step by step. The current executed line is number 62. The Watches frame at the right bottom of the window represents those variables which are watched when debugging.


dynamicform - [E:\syncFiles\CA7201\code\trunk\dynamicform] - [dynamicform-core] - ...\core\src\main\java\uk\ac\le\student\zl91\service\impl\QuestionManagerImpl...

```
57   public List<Question> findAvailableSubQuestion(Long qid
58       Map map = new HashMap();
59         map.put("qid", qid);
60       List<Question> questionList = this.getAll();
61       log.debug("All List.size():"+questionList.size());
62       Set <Question> subQuestionSet = new HashSet<Questio
63       for (Question question:questionList){
```

```
questionList.size() = 142
questionList = {java.util.ArrayList@11357} size = 142
  [0] = {uk.ac.le.student.zl91.model.Question@11359}"Question{qid=1, template=Template{tid=
    qid = {java.lang.Long@11565}"1"
    template = {uk.ac.le.student.zl91.model.Template@11566}"Template{tid=1, createTime=20
      tid = {java.lang.Long@11565}"1"
      createTime = {java.sql.Timestamp@13900}"2014-05-05 21:07:20.0"
      name = {java.lang.String@13901}"Report Foreign Body"
      status = {uk.ac.le.student.zl91.model.TemplateStatus@13902}"LIVING"
      questionSet = {org.hibernate.collection.internal.PersistentSet@13903} size = 15
    title = {java.lang.String@11567}"Site Name"
```

```
Frameworks detected: Hibernate, Spring, Struts 2, JPA frameworks are detected in the project Configure (yesterday 13:24)    63:40/12  CRLF  UTF-8
```

Figure 3-8-4: the screenshot of debugging software with break-points in IntelliJ IDEA

3.9 Deploy

For running the application, the following steps are required:

Download and install JDK7 and MySQL Server5.6, and then setup the development environment.

Import the data to MySQL server with the command as the following. (If an error shows up, try the file of code\trunk\deployed\zl91.db2)

```
                mysql -u <username> -h <hostname> -p <dbname> < zl91.db
For example: mysql -u root -h localhost -p dynamicform < zl91.db
```

Command segment 3-9-1: the command of MySQL for importing data

Unzip the file of code\trunk\deployed\apache-tomcat-7.0.53.dynamicform.zip, find the file of "apache-tomcat-7.0.53\webapps\ROOT\WEB-

INF\classes\applicationContext-resources.xml" and at the bottom of the file update the username and password with the account of root on MySQL database.

Run the file of startup.bat/startup.sh in the folder of apache-tomcat-7.0.53\bin. After "info: Server startup in xxxx ms/Tomcat started." shows up, open a browser and go to the address of 'localhost:8080'. Then there should be the sign in page shown up. Finally, login with the account of user name: 'dynamicform' and password: 'df'. The page of AppFuse QuickStart (http://appfuse.org/display/APF/AppFuse+QuickStart) may be helpful.

3.10 Testing

As mentioned above, there are two types of testing are required in the testing part. For fulfilled the requirement of testing, two tools are introduced. They are JUnit and Selenium[22]. JUnit is a test framework offering repeatable tests and based on Java language. The purpose of Selenium is for automatically testing web applications. Both of them not only can verify the functions against the requirements, but also offer a regressive test for preventing the software from introducing new bugs when developing new features or fixing existed bugs.

JUnit

Two test folders for the implementation of classes of DAO and Service are dynamicform\core\src\test\java\uk\ac\le\student\zl91\dao and dynamicform\core\src\test\java\uk\ac\le\student\zl91\service. All of the above tests are execute when running 'install' command with Maven. Alternatively other way of running a specific test class by running 'test' command followed the name of the test class. For instance, mvn test -Dtest=QuestionDaoTest.

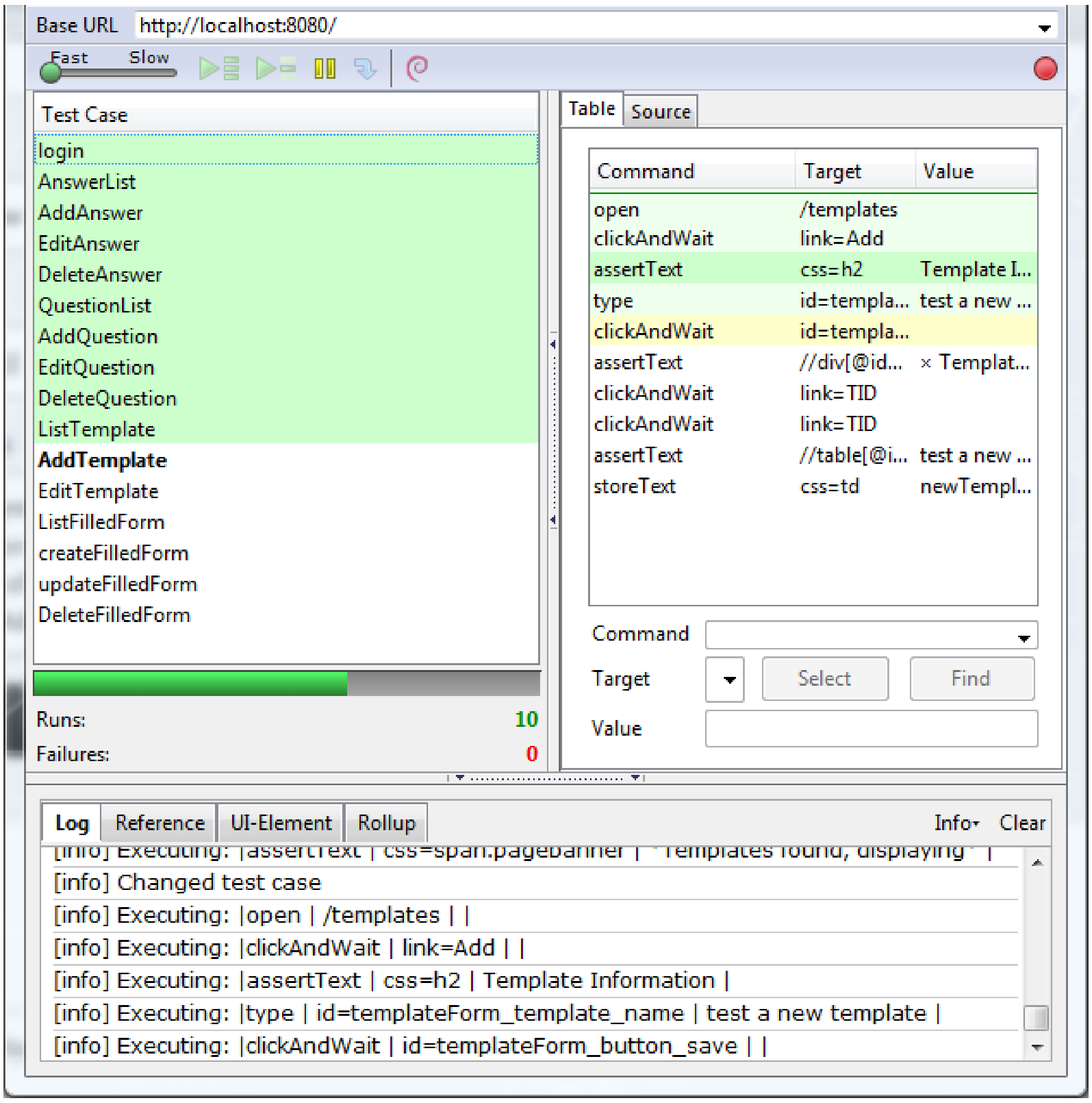
Another test class file of dynamicform\web\src\test\java\uk\ac\le\student\zl91\webapp\action\AnswerActionTest.java is for testing AnswerAction.java, but a bug in the Appfuse make the test

[22] SELENIUM, 2014-last update, Web Browser Automation. Available: http://docs.seleniumhq.org/ [4/30/2014, 2014]

fail before running it. The bug happens when the test framework tries to deleting all of the testing data from database.

Selenium

Selenium is a test tool for web application. There is a Firefox plug-in of simulate for user recording and playing back the actions on a web application. Once the test script files are saved, they are able to playback for regression test. A test suit may have a number of test cases. For testing this web application, one test suit of \code\trunk\test\TestCase\dynamicformTestSuit.html with sixteen test cases can load with the plug-in and then play back. The following is a screenshot shows the playback of test suit is in operation.



Base URL http://localhost:8080/

Test Case
login
AnswerList
AddAnswer
EditAnswer
DeleteAnswer
QuestionList
AddQuestion
EditQuestion
DeleteQuestion
ListTemplate
AddTemplate
EditTemplate
ListFilledForm
createFilledForm
updateFilledForm
DeleteFilledForm

Runs: 10
Failures: 0

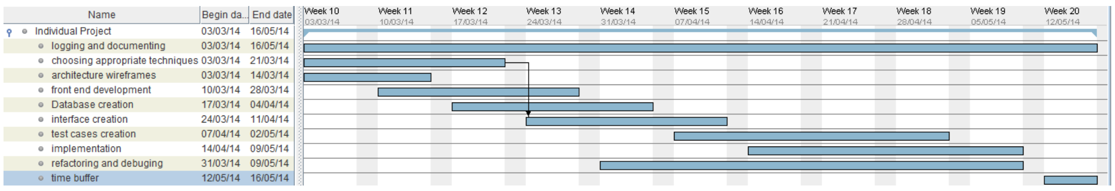
Command	Target	Value
open	/templates	
clickAndWait	link=Add	
assertText	css=h2	Template I...
type	id=templa...	test a new ...
clickAndWait	id=templa...	
assertText	//div[@id...	× Templat...
clickAndWait	link=TID	
clickAndWait	link=TID	
assertText	//table[@i...	test a new ...
storeText	css=td	newTempl...

```
[info] Executing: |assertText | css=span.pagebanner | * Templates found, displaying * |
[info] Changed test case
[info] Executing: |open | /templates | |
[info] Executing: |clickAndWait | link=Add | |
[info] Executing: |assertText | css=h2 | Template Information |
[info] Executing: |type | id=templateForm_template_name | test a new template |
[info] Executing: |clickAndWait | id=templateForm_button_save | |
```

Figure 3-10-1: the screenshot of running a test suit within Selenium

3.11 Plan

As ‘dynamic form’ software is developed by the author alone, the plan only includes two parts of time and tasks. The tool of GanttProject[23] 2.6.5 is introduced for drawing the time plan. The following screenshot is a Gantt chart from the file of \code\trunk\plan\timeplan.gan illustrates the time plan and tasks for the ‘dynamic form’ software.



Name	Begin da...	End date
Individual Project	03/03/14	16/05/14
logging and documenting	03/03/14	16/05/14
choosing appropriate techniques	03/03/14	21/03/14
architecture wireframes	03/03/14	14/03/14
front end development	10/03/14	28/03/14
Database creation	17/03/14	04/04/14
interface creation	24/03/14	11/04/14
test cases creation	07/04/14	02/05/14
implementation	14/04/14	09/05/14
refactoring and debuging	31/03/14	09/05/14
time buffer	12/05/14	16/05/14

Figure 3-11-1: the screenshot of time schedule of ‘dynamic form’

The time table of the project is shown as below.

Name	Begin date	End date	Duration(days)
Individual Project	03/3/2014	16/5/2014	55
logging and documenting	03/3/2014	16/5/2014	55
choosing appropriate techniques	03/3/2014	21/3/2014	15
architecture wireframes	03/3/2014	14/3/2014	10
front end development	10/3/2014	28/3/2014	15
Database creation	17/3/2014	04/4/2014	15
interface creation	24/3/2014	11/4/2014	15
test cases creation	07/4/2014	02/5/2014	20
implementation	14/4/2014	09/5/2014	20
refactoring and debugging	31/3/2014	09/5/2014	30
time buffer	12/5/2014	16/5/2014	5

Figure 3-11-2: the table of time schedule of ‘dynamic form’

[23] 2014-last update, Gantt Project. Available: http://www.ganttproject.biz/ [5/10/2014, 2014]

Chapter 4 Management of Inviso Incident

I have to claim that the 'Inviso Incident' project had been initialised when I commenced the cooperation with ESB. At the meanwhile, some of resources have been ready for the requirement phase of the 'Inviso Incident' project. Since that time, I had been involved in the 'Inviso Incident' project.

4.1 Circumstance investigation

Background

The project of 'Inviso Incident' is for building a web-based incident reporting and investigation software for ESB. ESB expect that the software can both cover the requirements from CH & Co and have the ability of expansion for tackling various structures of forms. ESB currently utilise a third part online application to help the incident reporting and investigation. But they expect to have their own management system which is designed for integrating into ESB's business and help ESB promote competitive advantage in the marketplace of consultancy. Besides the advantage, it includes:

- As the owner of the software, ESB can extend more functionality for adapting the improvement of their business.
- As the system does not rely on a third part database, it becomes more confidence of the data.
- ESB could gather statistic information from their clients and give the average level of incident rate.
- Tailoring the services according to various requirement from clients

Stakeholders

"Stakeholders include all members of the project team as well as all interested entities that are internal or external to the organization."(Project Management Institute 2013) The following list illustrates those stakeholders recognised in the project:

- The project sponsor is ESB Company.
- Project manager who take responsibility for the project is me.

- Specialists who know the requirements of the software are two staff from ESB.
- A user interface (UI) designer who develops the web UI for the software is a freelancer.
- Requirement analysis who transforms the requirements to prototypes is me.
- An outsourcing company who takes the responsibility for the software implementation is called Drupal from India.
- A vender who provides the Short Message Service (SMS) for the software is unspecified.
- A vender who provides the hosting service for the web application is specified.
- The clients who pay for it. One of the clients is CH & Co.
- The end users who use the software are from ESB and CH & Co.

4.2 Scope management

Collect requirements

Most requirements come from two specialist of ESB. They are Bob Mackay and Kirstie Davidson who have many years' experiences in the fields of training, food safety, health & safety and fire safety consultancy. They are not only master the regulations and rules announced by authorities, but also good at the risk assessment of all kinds of hazards in business. Another source of requirement is from the authorities called Health and Safety Executive[24] (HSE).

The process of collecting requirement, referring to the Volere prototyping activity[25] illustrates as following steps:

[24] HSE, 2014-last update, Information about health and safety at work. Available: http://www.hse.gov.uk/ [4/29/2014, 2014]

[25] VOLERE, 2014-last update, The (proto)type of requirements. Available: http://www.volere.co.uk/reqsatiag.htm [5/1/2014, 2014]

- The first step: according to the SOW and background of the project, I email the specialists ask for the specification of incident report forms and incident investigation forms.
- The second step: after they study and work out those forms, and then reply me with their outcomes.
- The third step: I transform them into prototypes, and ask them for a confirmation.

Analyse requirements

Those documents are verified and refined with the perspective of information technology. The consideration and judgement on the possibility and cost of those specific requirements comes from the experiences of former projects I have achieved. The general modules of 'Inviso Incident' shows as following:

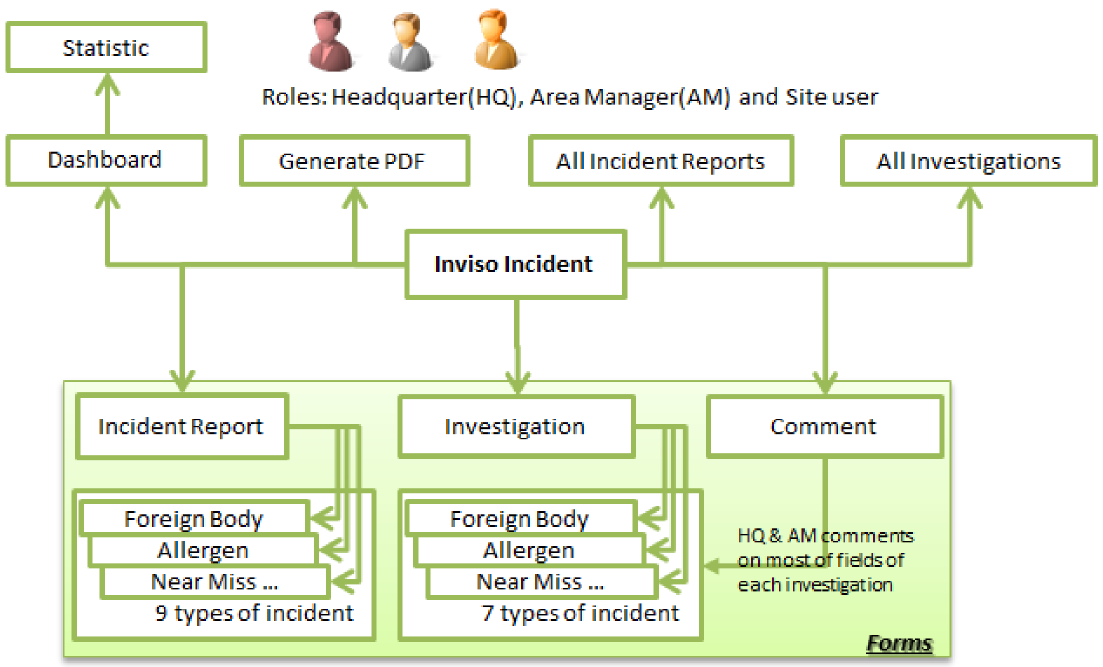


Figure 4-2-1: the modules of 'Inviso Incident'

- There are three roles Headquarter (HQ), Area Manager (AM) and Site user in this system.
- Three major business functions base on form within this software. They are:
    - Report an incident by filling out various types of report form

- Investigate an incident by filling out various types of investigation form
- Comment the forms above

When the incidents take place on a premise, the site user or area manager from the clients fill out a form for reporting the incident to its area manager or headquarter. All of the roles can review and update reports and investigations belong to their own or subordinate. They can, moreover, get Accident Frequency Rate (AFR) and Accident Incident Rate (AIR) of their organizations. If some incident applies to injures of Reporting of Injuries, Diseases and Dangerous Occurrences Regulations (RIDDOR), the incident report are allowed to export as a PDF format file.

The requirements come from SOW and specialist. All of the requirements can be divided into functional and non-functional.

Functional requirements include the following items:

- User management. The development of this model does not belong to the project. However this project relies on the user management model which has integrated with ESB web site. The roles shown as following.

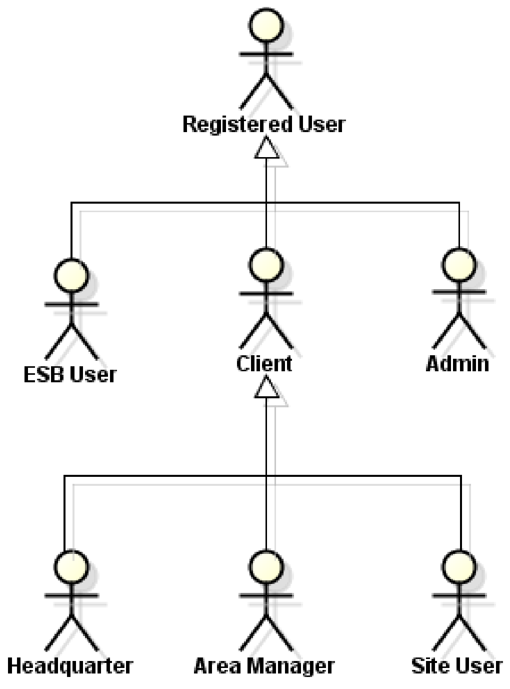


Figure 4-2-2: the roles in the 'Inviso Incident'

- Registered User. This is a general role of all users.
- Admin. This role should be implemented in the ESB web site. Admin can create, delete and modify the roles of headquarter, area manager, user, and entities of organisation, brand, site.
- ESB User. Those people are staff of ESB and have privilege to view all reports and investigations of their clients.
- Client User. Three types of role for client user are Headquarter, Area Manager and Site User. The site user refers to a site manager who manages an individual catering operation or unit. The area manager refers to a senior manager who is responsible for several individual catering operation or unit. The headquarter represents the head office which manages all managers and individual units.

- Organisation management. It should be implemented by Audit software. Since the Audit software has the same requirement for the organisation management, the 'Inviso Incident' can use it directly.
  - Create, modify, delete, list and review organisations.
  - Assign a site to an area manager and assign an area manager to headquarter.
- Incident report management
  - Create, modify, list, review and search incident reports
  - Export incident reports to PDF format files
  - Support RIDDOR and non-RIDDOR incident report
  - Nine types of report are supported. They are foreign body, chemical contamination, allergen, suspected food poison, accident, near miss, dangerous occurrence, environment incident and enforcement officer visit.
- Incident investigation management
  - Create, modify, list, review, search and comment incident investigations.

- Severn types of investigation are supported. They are allergen, foreign body, chemical contamination, suspected food poison, accident, dangerous occurrence and environment incident.

- Dashboard.
  - A histogram of last quarter number of incident. This histogram illustrates the total number of incidents happened in last four months respectively.
  - A table for rate comparison. This table represents the AFR and AIR for three ranges which are a specified organisation, the company and the whole industry.

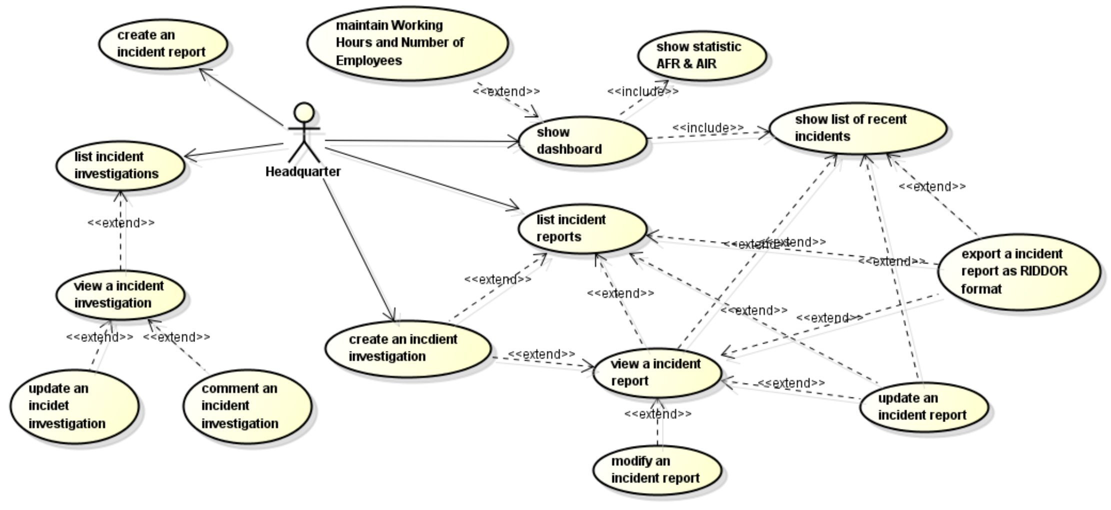

create an incident report
maintain Working Hours and Number of Employees
show statistic AFR & AIR
Headquarter
list incident investigations
show dashboard
show list of recent incidents
list incident reports
view a incident investigation
create an incdient investigation
export a incident report as RIDDOR format
view a incident report
update an incidet investigation
comment an incident investigation
update an incident report
modify an incident report
<<extend>>
<<include>>

Figure 4-2-3 :The user cases for the role of headquarter

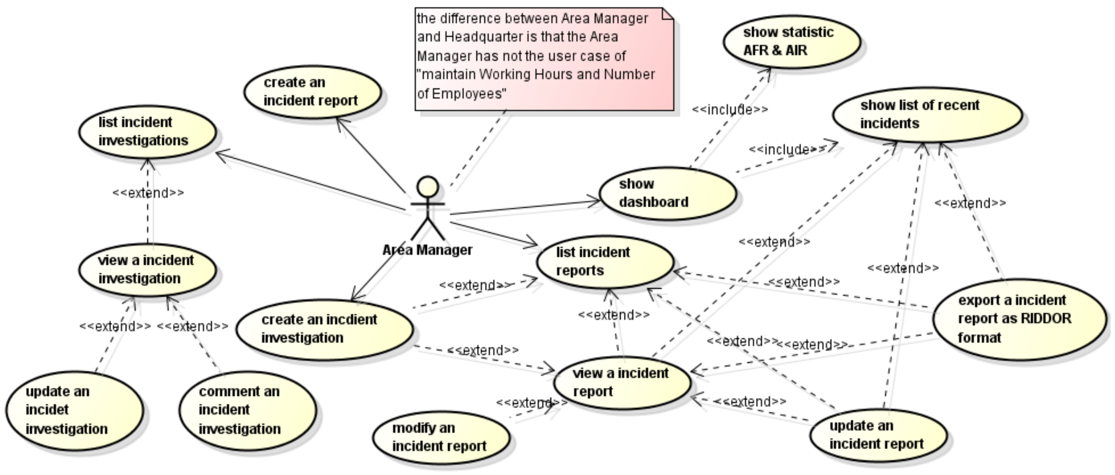

the difference between Area Manager and Headquarter is that the Area Manager has not the user case of "maintain Working Hours and Number of Employees"
show statistic AFR & AIR
create an incident report
show list of recent incidents
list incident investigations
show dashboard
Area Manager
list incident reports
view a incident investigation
create an incdient investigation
export a incident report as RIDDOR format
view a incident report
update an incidet investigation
comment an incident investigation
modify an incident report
update an incident report
<<extend>>
<<include>>

Figure 4-2-4 :The user cases for the role of area manager

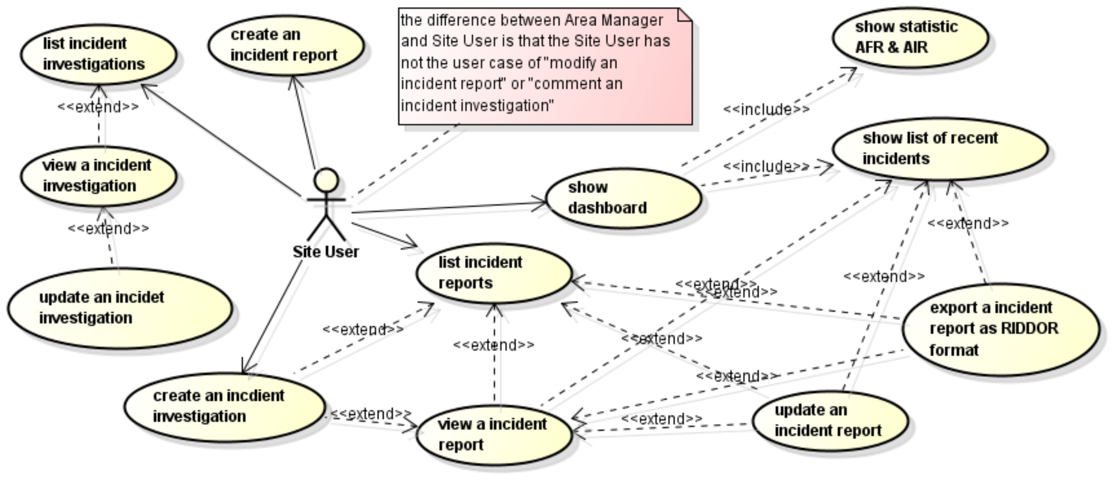

list incident investigations
create an incident report
the difference between Area Manager and Site User is that the Site User has not the user case of "modify an incident report" or "comment an incident investigation"
show statistic AFR & AIR
<<include>>
show list of recent incidents
<<extend>>
view a incident investigation
Site User
show dashboard
update an incidet investigation
list incident reports
export a incident report as RIDDOR format
create an incdient investigation
view a incident report
update an incident report

Figure 4-2-5 :The user cases for the role of site user

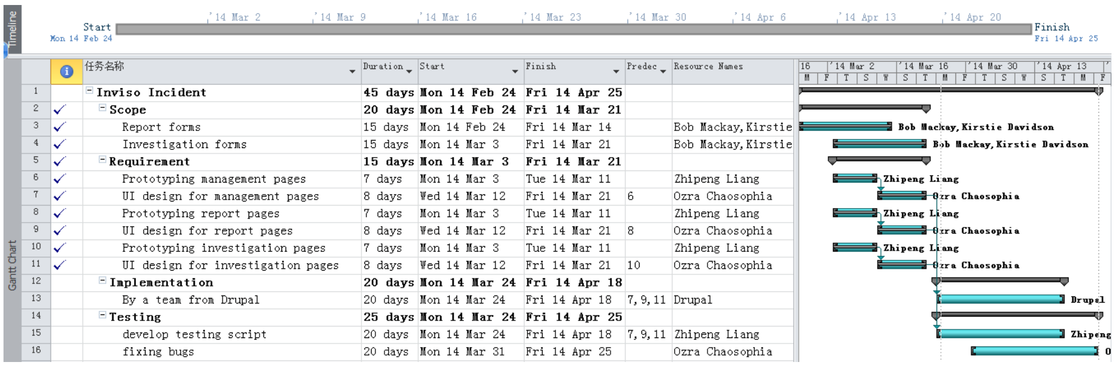
The list of Non-functional requirements is represented below:

- Accessibility. The software is supposed to be accessed form PCs and mobile devices (e.g. tablet). The solution is Bootstrap.
- Compatibility. The popular web browsers are supported such as Internet Explorer, Firefox and Safari. In addition, the database is required to be compatible to MySQL.
- Availability. The software is supposed to offer 24 x 7 online services. The interruption for routine update is acceptable.
- Response time. The page should have a reasonable response times. Most page loadings should be less than 2 seconds. Some pages with heavy computing tasks should complete loading in 4 seconds.
- Safety. Firewall and anti-virus software/hardware are required for protecting the online service from the damage of hack attacking or virus.
- Supportability. A technical support within normal working hours is required, which is conducted by the outsourcing company.
- Useable. Good UI design and efficient to use by target users.

4.3 Time management

Before draw the time schedule, it is necessary to create Work Breakdown Structure (WBS). 'Create WBS is the process of subdividing project deliverables and project work into smaller, more manageable components.' (Project Management Institute

2013: 125) According to the previous experience form project management, the specific numbers of working days assign to each leaf tasks. The total days of high level tasks is the sum of its sub-tasks' days. In addition, some tasks may depend on others, which means one task cannot start until one or more tasks is completed. Besides the constraints between tasks, the resource affects the tasks as well. A task may not commence unless it get the critical resource. The resource may be a people, a machine or a fund. For drawing the time schedule of this project, all above factors has to be taken into account. For clearly representing the time schedule with WBS and its depended resource, MS Project is introduced. The following screenshot from the file of \code\trunk\plan\InvisoIncient.mpp illustrates the time schedule with WBS and its depended resource.



		任务名称	Duration	Start	Finish	Predec	Resource Names
1		**Inviso Incident**	**45 days**	**Mon 14 Feb 24**	**Fri 14 Apr 25**		
2	✓	**Scope**	**20 days**	**Mon 14 Feb 24**	**Fri 14 Mar 21**		
3	✓	Report forms	15 days	Mon 14 Feb 24	Fri 14 Mar 14		Bob Mackay,Kirstie
4	✓	Investigation forms	15 days	Mon 14 Mar 3	Fri 14 Mar 21		Bob Mackay,Kirstie
5	✓	**Requirement**	**15 days**	**Mon 14 Mar 3**	**Fri 14 Mar 21**		
6	✓	Prototyping management pages	7 days	Mon 14 Mar 3	Tue 14 Mar 11		Zhipeng Liang
7	✓	UI design for management pages	8 days	Wed 14 Mar 12	Fri 14 Mar 21	6	Ozra Chaosophia
8	✓	Prototyping report pages	7 days	Mon 14 Mar 3	Tue 14 Mar 11		Zhipeng Liang
9	✓	UI design for report pages	8 days	Wed 14 Mar 12	Fri 14 Mar 21	8	Ozra Chaosophia
10	✓	Prototyping investigation pages	7 days	Mon 14 Mar 3	Tue 14 Mar 11		Zhipeng Liang
11	✓	UI design for investigation pages	8 days	Wed 14 Mar 12	Fri 14 Mar 21	10	Ozra Chaosophia
12		**Implementation**	**20 days**	**Mon 14 Mar 24**	**Fri 14 Apr 18**		
13		By a team from Drupal	20 days	Mon 14 Mar 24	Fri 14 Apr 18	7,9,11	Drupal
14		**Testing**	**25 days**	**Mon 14 Mar 24**	**Fri 14 Apr 25**		
15		develop testing script	20 days	Mon 14 Mar 24	Fri 14 Apr 18	7,9,11	Zhipeng Liang
16		fixing bugs	20 days	Mon 14 Mar 31	Fri 14 Apr 25		Ozra Chaosophia

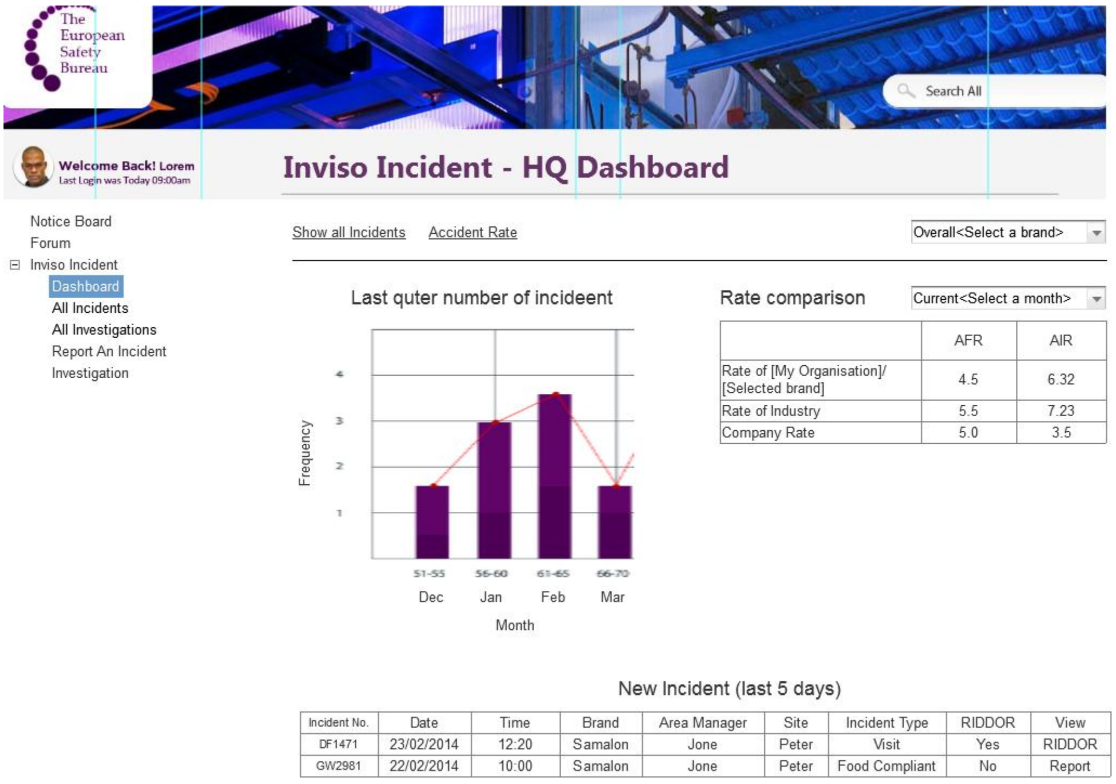
Figure 4-3-1 :The time schedule with WBS of 'Inviso Incident' project

The tick marks on the right side suggest those tasks are completed. The horizontal black lines in the middle of the tasks represent how much those tasks have been completed.

4.4 Design of prototyping

Since the implementation has been outsourced to the Indian company, my task of the design includes two parts: prototype design and database design.

Axure is introduced for conducting the prototype design. Axure is an easy-use and powerful prototyping tool. At first, it involves the basic HTML elements. By simple operations like drag and drop, developers are able to effectively design a web page. Secondly, it allows developers to create dynamic elements on the prototype. Therefore those HTML elements can response the action from user. For example, a user select the last answer item of ‘Other’, then the text field with a label of ‘Please specify’ shows up. Last but not least, the prototypes made by Axure can be export to images and HTML. In other words, the developers can get the static or dynamic outcomes. In this project the most prototype pages are made by Axure. However comparing to a sketch, the drawback of Axure is a time consuming for prototyping a big form. For that reason I did some sketch on paper for catching up the time schedule. The following is one of the prototyping pages of headquarter’s dashboard from the file of \code\trunk\prototype\incidnet7.0.rp.



Inviso Incident - HQ Dashboard

Last quter number of incideent

Rate comparison

	AFR	AIR
Rate of [My Organisation]/ [Selected brand]	4.5	6.32
Rate of Industry	5.5	7.23
Company Rate	5.0	3.5

New Incident (last 5 days)

Incident No.	Date	Time	Brand	Area Manager	Site	Incident Type	RIDDOR	View
DF1471	23/02/2014	12:20	Samalon	Jone	Peter	Visit	Yes	RIDDOR
GW2981	22/02/2014	10:00	Samalon	Jone	Peter	Food Compliant	No	Report

Figure 4-4-1 : the prototyping page of headquarter’s dashboard

Beside those prototyping pages above, there are some sketches of prototype design which are scanned in the folder of \code\trunk\prototype\sketch.

Chapter 5 Conclusion

With the great help from Irek, Reiko and ESB, the author has completed all objects of the two tasks within the duration of individual project and gained fruitful experience on both sides of academy and industry. Not only has the skills of software development been improved, but also the management knowledge and skills have been applied into the workplace.

Given more time, those features below could be developed as well.

- Page split. For some complex form, organising the content with split pages is better than put all on one page.
- Scoring. This feather is designed for audit and examination. Users are allowed to assign a specific score to each answer or question when they are building up a template. After those templates are filled out, the total score are counted for the current user.
- Customising the style and theme of template. For commercialising and offering the maximising user experience, all the elements on the form, including background colour, logo, menu, the head and foot of the page, are allow customised by users.

The original expectation of the management of 'Inviso Incident' software includes five parts. Except of the part of testing and evaluation, the rest parts are achieved. The main reasons of missing the last part are below.

- Time is not enough. According to the time schedule, the testing phase is compressed and heavily overlapped with the phase of development.
- Many uncertain factors affect and slow down the progress of project. The customers not participating into the requirement phase at the most begging is one of key factors which slow down the project.
- The problem of communication. As the people of the project do not work in a place, it makes a serious delay when the information and instructions passing.

For fixing the problem above or preventing them happening, risk management and communication management should be introduced in the process of project.

Bibliography

BRUCE ECKEL, 2006. *Thinking in Java.* 4th edn. Prentice Hall.

COLE, K., MCCHESNEY, R. and RASZKA, R., 2011. *Advanced Java EE development for Rational Application Developer 7.5: developers' guidebook.* 1 edn. Ketchum, Idaho: MC Press.

GONCALVES, A., 2013. *Beginning Java EE 7.* APRESS.

GUPTA, A., 2013. *Java EE 7 essentials.* Sebastopol, CA: O'Reilly Media.

JENDROCK, E., CERVERA-NAVARRO, R., EVANS, I., HAASE, K. and MARKITO, W., 2014-last update, The Java EE 7 Tutorial. Available: http://docs.oracle.com/javaee/7/tutorial/doc/home.htm [04027, 2014].

MARCOTTE, E., 2014-last update, Responsive Web Design. Available: http://alistapart.com/article/responsive-web-design [5/9/2014, 2014].

ORACLE, 2014-last update, Distributed Multitiered Applications. Available: http://docs.oracle.com/javaee/7/tutorial/doc/overview003.htm#BNAAY [05/02, 2014].

PROJECT MANAGEMENT INSTITUTE, 2013. *A guide to the project management body of knowledge (PMBOK® guide).* 5th edn. Newtown Square, Pa.: Project Management Institute.

RAIBLE, M., 2014-last update, AppFuse QuickStart. Available: http://appfuse.org/display/APF/AppFuse+QuickStart [5/4/2014, 2014].

ROBERTSON, S. and ROBERTSON, J., 2006. *Mastering the Requirements Process Second Edition.* 2nd edn. Indiana: Addison Wesley Professional.

SEDDIGHI, A.R., 2009. *Spring Persistence with Hibernate build robust and reliable persistence solutions for your enterprise Java application.* Birmingham, U.K.: Packt Pub.