3-2017

# AUTOMATIC GENERATION OF WEB APPLICATIONS AND MANAGEMENT SYSTEM

Yu Zhou
004306704@coyote.csusb.edu

# AUTOMATIC GENERATION OF WEB APPLICATIONS AND MANAGEMENT SYSTEM

---

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Yu Zhou

March 2017

# AUTOMATIC GENERATION OF WEB APPLICATIONS AND MANAGEMENT SYSTEM

---

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Yu Zhou

March 2017

Approved by:

Dr. Zhengping Wu, Advisor, Computer Science and Engineering

Dr. George Georgiou, Committee Member

Dr. Owen J. Murphy, Committee Member

## ABSTRACT

One of the major difficulties in web application design is the tediousness of constructing new web pages from scratch. For traditional web application projects, the web application designers usually design and implement web application projects step by step, in detail. My project is called “automatic generation of web applications and management system.” This web application generator can generate the generic and customized web applications based on software engineering theories. The flow driven methodology will be used to drive the project by Business Process Model Notation (BPMN). Modules of the project are: database, web server, HTML page, functionality, financial analysis model, customer, and BPMN. The BPMN is the most important section of this entire project, due to the BPMN flow engine that most of the work and data flow depends on the engine. There are two ways to deal with the project. One way is to go to the main page, then to choose one web app template, and click the generating button. The other way is for the customers to request special orders. The project then will give suitable software development methodologies to follow up. After a software development life cycle, customers will receive their required product.

TABLE OF CONTENTS

LIST OF FIGURES

# CHAPTER ONE

## INTRODUCTION

One of the major difficulties in web application design is the tediousness of constructing new web pages from scratch. For traditional web application projects, customers usually need to find software companies who can design web application and follow software development methodologies to design customers' web applications in the past few years. Recently, many online web application generators have appeared. Weebly and WordPress (online web application builder tool) have become the most used online web project generators in the U. S., but their customers still need to modify their projects such as re-layout pages and add additional elements to their web applications. The aim of the project is to make it more user-friendly for customers, who not familiar the general technology of web application. This project is named "Automatic Generation of the Web Applications." The project will use flow driven methodology to generate web applications. Based on the flow driven methodology, Business Process Model Notation (BPMN) is the top choice. The project will also include the software engineering theories because some ways of generating web applications are for customers to submit their special requirements on customized project order pages. The project will use suitable software development methodologies to match customers' web application projects. After a software development life cycle, customers will receive their required product and then they need to add them to their projects.

# CHAPTER TWO

## DEVELOPMENT TOOLS AND ENVIRONMENTS

There are four main development tools that I used them to design and develop my project and web service. First, Eclipse is used to develop the project and web service. Second, Apache Tomcat is used as a web host environment for the project and web service. Third, MySQL is used as a data store where data will be saved into. The last one is UML editor that will be used to model a relationship between database and project components.

Eclipse, which is an integrated development environment (IDE) and what is the most widely used Java IDE, offers a base workspace and an extensible plug-in component for customizing development environment [19]. Because of the Windows-based computer, third party plug-ins can be easily embedded into Eclipse development environment and assistant Java web developers to develop project such as SVN. The eclipse also provides powerful support for development and testing of the project and web services, containing the capability of debugging, inspect variables and step through the code.

Apache Tomcat is an open-source and widely used Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, Java Server Pages (JSP). On Eclipse platform, the tomcat web service can be controlled by the Eclipse IDE [20].

MySQL is an open-source relational database management system (RDBMS). Eclipse provides many database interfaces including MySQL and it is easy to setup on the Windows-based environment to produce the project database by using Eclipse database tool [21].

In the field of software engineering, the UML stands for Unified Modeling Language, which is a general-purpose, developmental and modeling language. The UML is intended to provide a standard way to visualize the design of a project here [22].

# CHAPTER THREE

# IMPLEMENTATION

## Implement Plan

To design the automatic generation of web application and management system project, I will use the spiral software development model, which is a risk-driven software process framework [7]. At the first development life cycle, I will finish 40% functionalities of the total project. The 40% functionalities of the whole project will include the BPMN, which is a systemic approach for capturing, designing, executing, documenting, measuring, monitoring, and controlling both automated and non-automated processes to meet the objectives and business strategies of a company [3].

The BPMN can offer most of the flexible workflows and three static logic flows. Due to the BPMN technology, the workflow can be generated and modified by BPMN flow tool. In my project, I will use this technology for the web page workflow. For logic flows, I will use the flow controller to control these three static logic flows. They are online shopping logic flow, online hotel logic flow, and online restaurant reservation flow. According to the BPMN's workflow, the flow controller can decide what page and what page's data should be pop out when customers make requirements on that web page. Customers can modify the workflow at Customer’s management platform. According to the technology I decide to use, the project's construction is also important. All web pages and

functions will be decomposed into code fragments and save into database. Html and jQuery will make the web pages more flexible and user-friendly.

BPMN can also guide the customers to finish their first web application, web application data controller (where customers and system administrators can control and manage data at different interfaces), Email controller as a notification functionality that will be embedded into web applications, web server controller (where customers and system administrators can control the tomcat at different interfaces), administration platform (where system administrators can manage the system's configuration, web application's template, functionality, and database.) The Database controller can be also used and managed by customers at customer's interfaces (where customers only can see the data and manage the data rather than the database itself). To design this project, I use Java 7.0 as the basic development language, BPMN as work and logic flow controller, Html [6], jQuery [9], Tomcat [5], Apache [2], Java SSH (Spring, Hibernate, and Struts) framework [1][8] and Maven together [10]. On web application page, the web application generator can offer these elements that they are text boxes, area text box, submit button, reset button, customized button, search area, single image and slide image box, text label, sub window, and section area.

## Libraries and Techniques

### Maven

I chose to use Maven for my project. That was because the Wikipedia said "Maven was originally started as an attempt to simplify the build processes in the Jakarta Turbine project. There were several projects each with their own Ant build files that were all slightly different and JARs were checked into CVS [17]". The primary gold of Maven is to accept the Java developers to comprehend the whole state of the development effort with a short period [17]. The Wikipedia also said that "in order to attain this goal there are several areas of concern that Maven attempts to deal with:

Making the build process easy.

Providing a uniform build system.

Providing quality project information.

Providing guidelines for best practices development.

Allowing transparent migration to new features [17]."

The Maven was used as a project construction management tool to implement the project because the Maven can offer me an overview of my project construction such as collection of java jars and embedded plug-ins tomcat. If I choose not use the Maven, then I should upload and import library references to the project. If I change the project workspace path, I must re-point the library references to the project.

MVC Framework

Model–view–controller (Figure 3.1) is a software design pattern for implementing customer interfaces on computers. It divides a given software application into three interconnected parts [18]. To connect each component between model and controller, spring framework and java annotation was used to connect them. When a model is declared, then I should add a java entity annotation and put it above a class. To separate internal representations of information from the ways that information is presented to or accepted from the customers. The model directly manages the data, logic, and rules of the application. The view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The third part, the controller, accepts input and converts it to commands for the model or view [18].

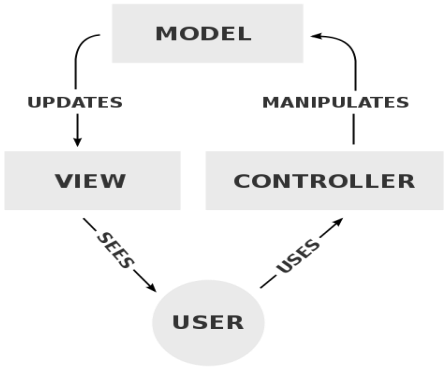



Figure 3.1. MVC.

Spring Framework

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform [11]. Although the spring framework does not include any specific programming model, it has become popular and been using widely in the Java community.

The spring framework can provide a comprehensive programming and configuration model for my Java-based web applications. Based on this benefit, the spring framework allows me to embedded other components easily such as Hibernate, BPMN engine, MVC design pattern. To embed another component into spring, I need to make other components as objects and then inject them into spring XML file.

Hibernate Framework

Hibernate ORM (hibernate in short) is an object-relational mapping tool for the Java language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions [14].

The main hibernate's feature is mapping from Java classes to database tables, and mapping from Java data types to database data types. Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual handling and object conversion of the result set.

Using the hibernate framework to manage database is easier than using direct database connection method because the hibernate framework will help me to map java entity elements with database table elements by add annotation at each entity class and inject them into hibernate configuration file. If I use direct database connection method to implement the project, I must implement the database connection method and connection pool, execute queue, and map data manually.

BPMN

Business Process Model and Notation (BPMN) is a graphical representation for specifying business processes in a business process model.

Business Process Management Initiative (Figure 3.2) developed BPMN, which has been maintained by the Object Management Group since the two organizations merged in 2005. Version 2.0 of BPMN was released in January 2011, at which point the name was adapted to Business Process Model and Notation as execution semantics were also introduced alongside the notational and diagramming elements [16].

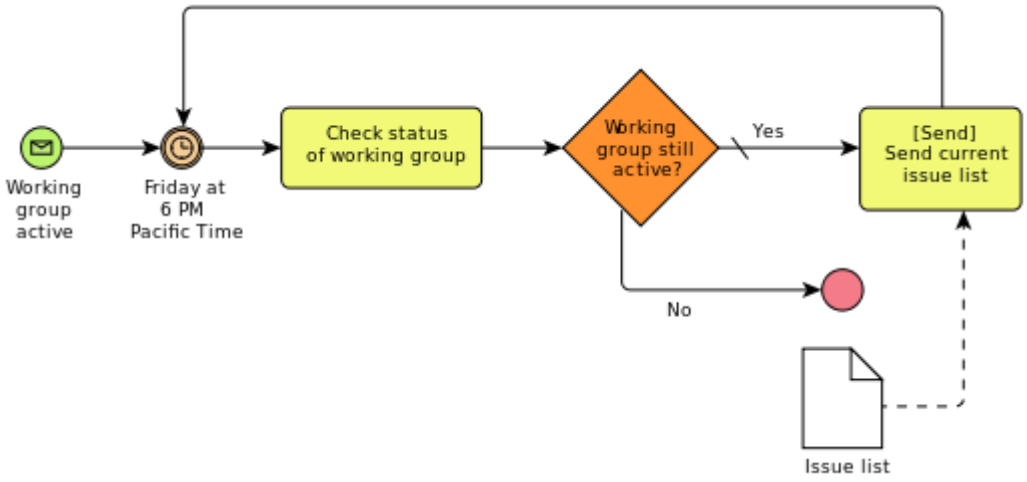



Figure 3.2. Table of country.

Because BPMN can be embedded into spring framework, the whole workflow logic is not needed when implementing workflow component. What I

need to do is to design a BPMN process chart and some extra java function or js functions that the process may use.

SOAP

SOAP (Simple Object Access Protocol) is a protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to induce extensibility, neutrality, and independence. It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. SOAP allows processes running on disparate operating systems (such as Windows and Linux) to communicate using Extensible Markup Language (XML). Since Web protocols like HTTP are installed and running on all Operating systems, SOAP allows clients to invoke web services and receive responses independent of language and platforms [12].

This project has a subsystem to work with the main system. That system is a file system which handles all files and folder operation, including creation, duplication, deletion, and update. The main system will send a command with parameters to the subsystem. If it need to operate files such as copying original tomcat files, updating tomcat files, and uploading file from the customer system to the service. Eclipse was used to add web service and web client to these systems. All transactions are byte data type and in JSON style.

REST

Representational State Transfer(REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined the set of stateless operations [13]. Using HTTP, the kind of operations available include those predefined by the HTTP verbs GET, POST, PUT, DELETE and so on.

REST is used as the connection between the main system and BPMN engine. The REST has less work to do than the SOAP, and the REST will return a JSON string to the client and display the content on the customer screen. The SOAP should have a client side as a message receiver, but the REST does not need. So, the REST can save more work than developing a SOAP connection.

jQuery Ajax

The Wikipedia recommended me to use the jQuery for my project front side. The reason is “jQuery (Figure 3.3) is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. jQuery is the most popular JavaScript library in use today, with installation on 65% of the top 10 million highest-trafficked sites on the Web. jQuery is free, open-source software licensed under the MIT License [15].”

Ajax is a set of web development techniques using many web technologies on the client-side to create asynchronous Web applications. With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

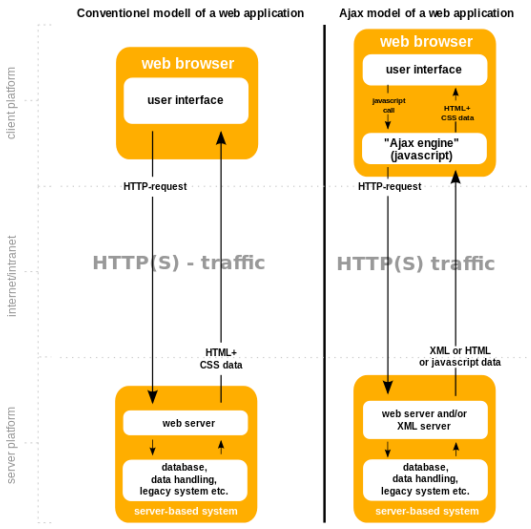



Figure 3.3. jQuery ajax work flow.

# CHAPTER FOUR

# PROJECT CONSTRUCTION

## Use Case Diagrams

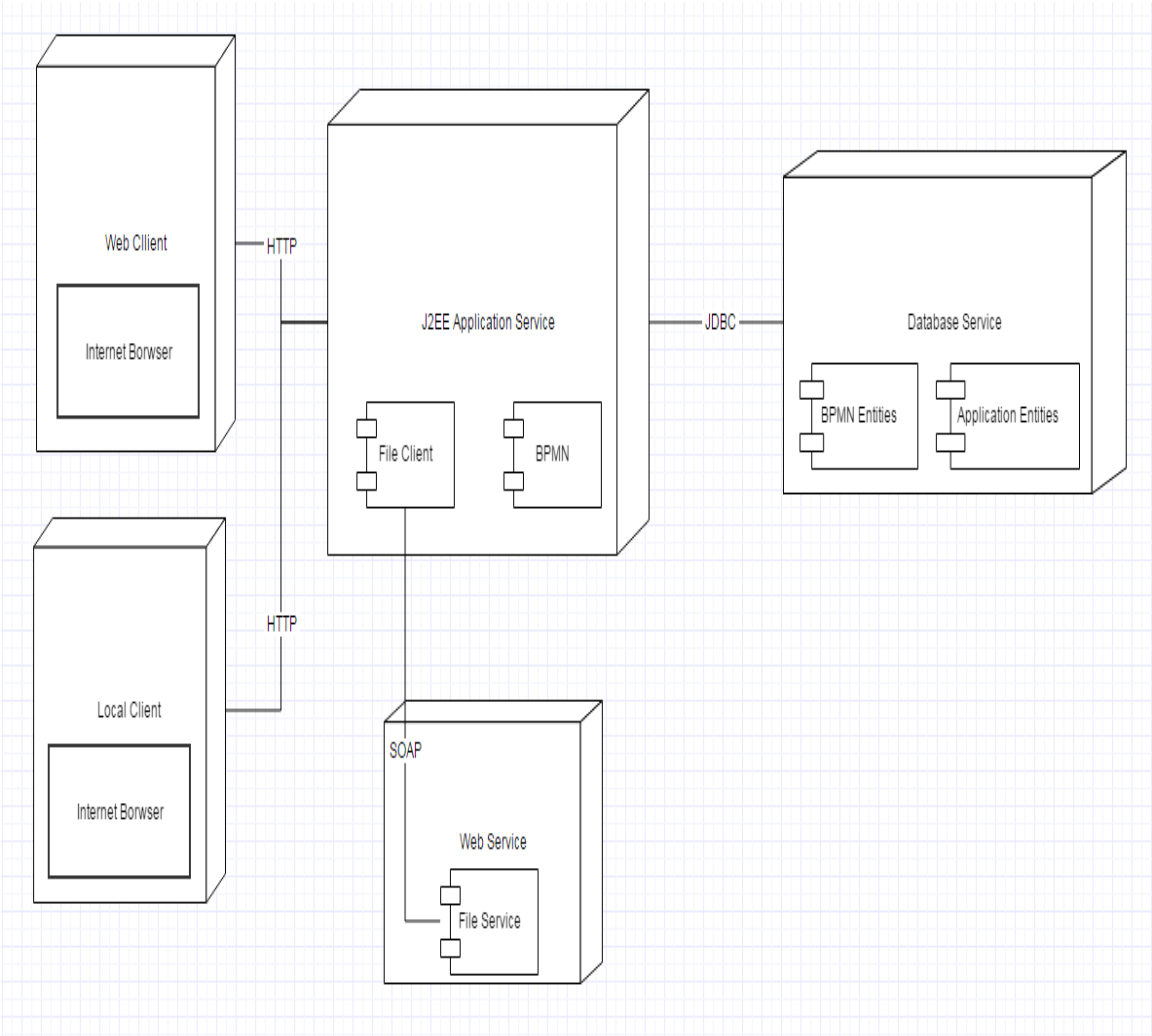



Figure 4.1. Overview of system construction.

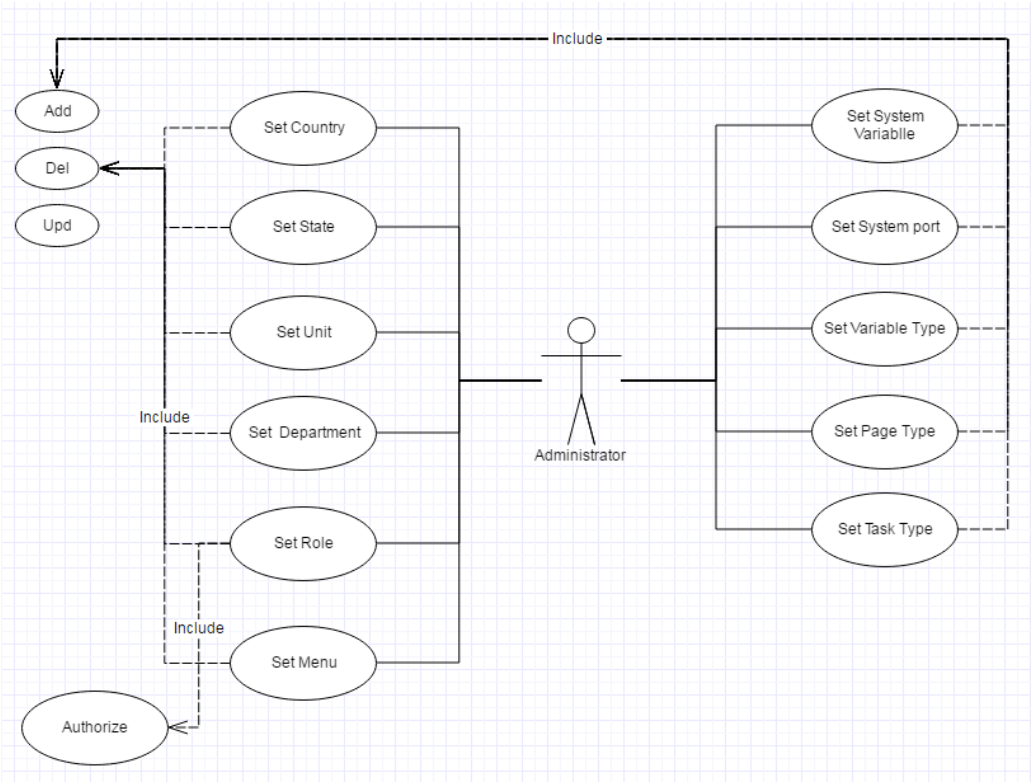



Figure 4.2. System configuration component use case.

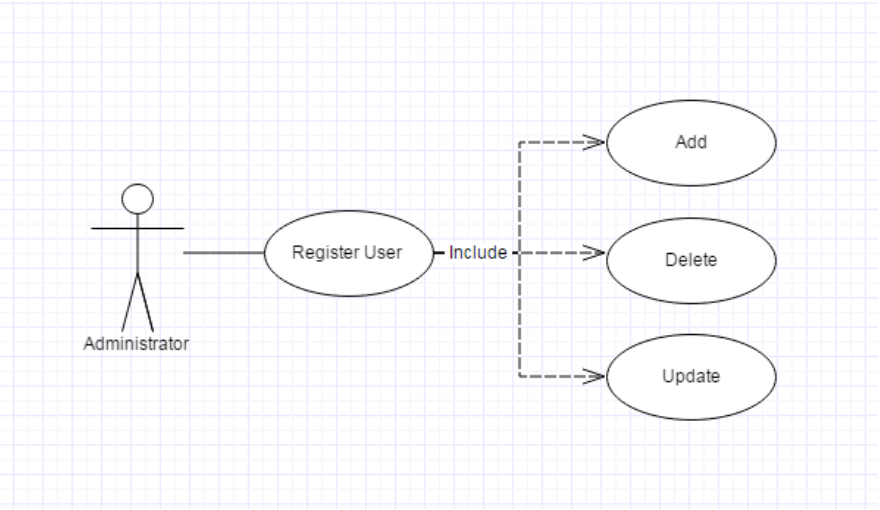



Figure 4.3. Customer component use case.

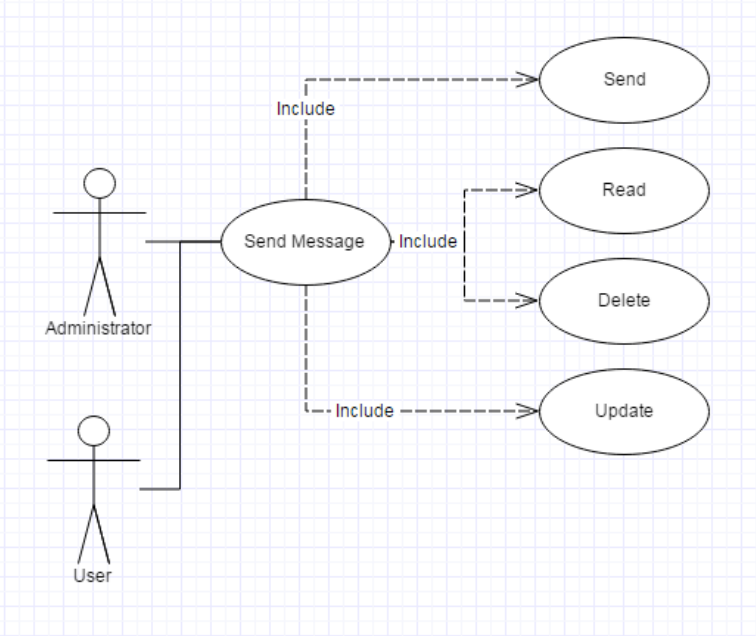



Figure 4.4. Message component use case.

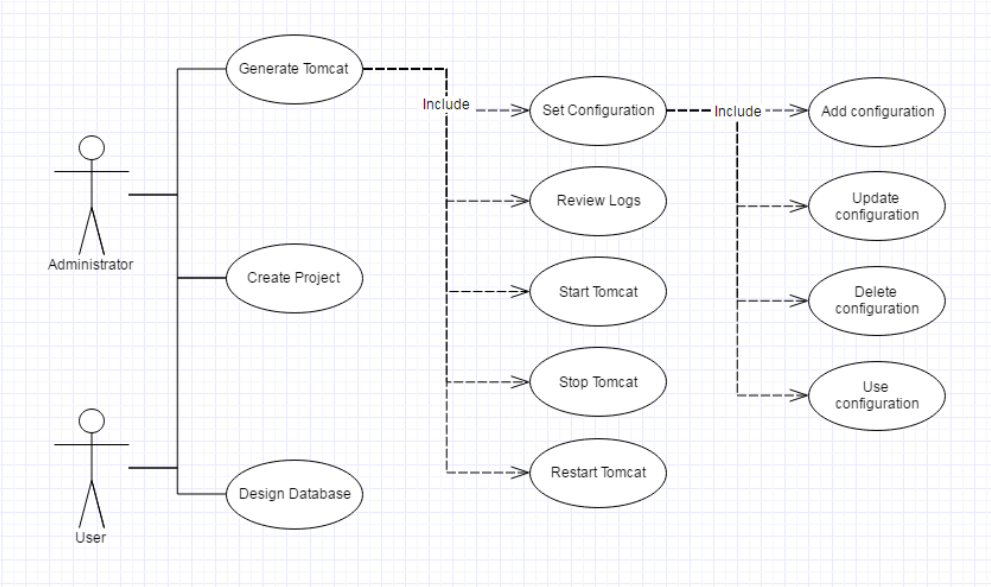



Figure 4.5. Tomcat component use case.

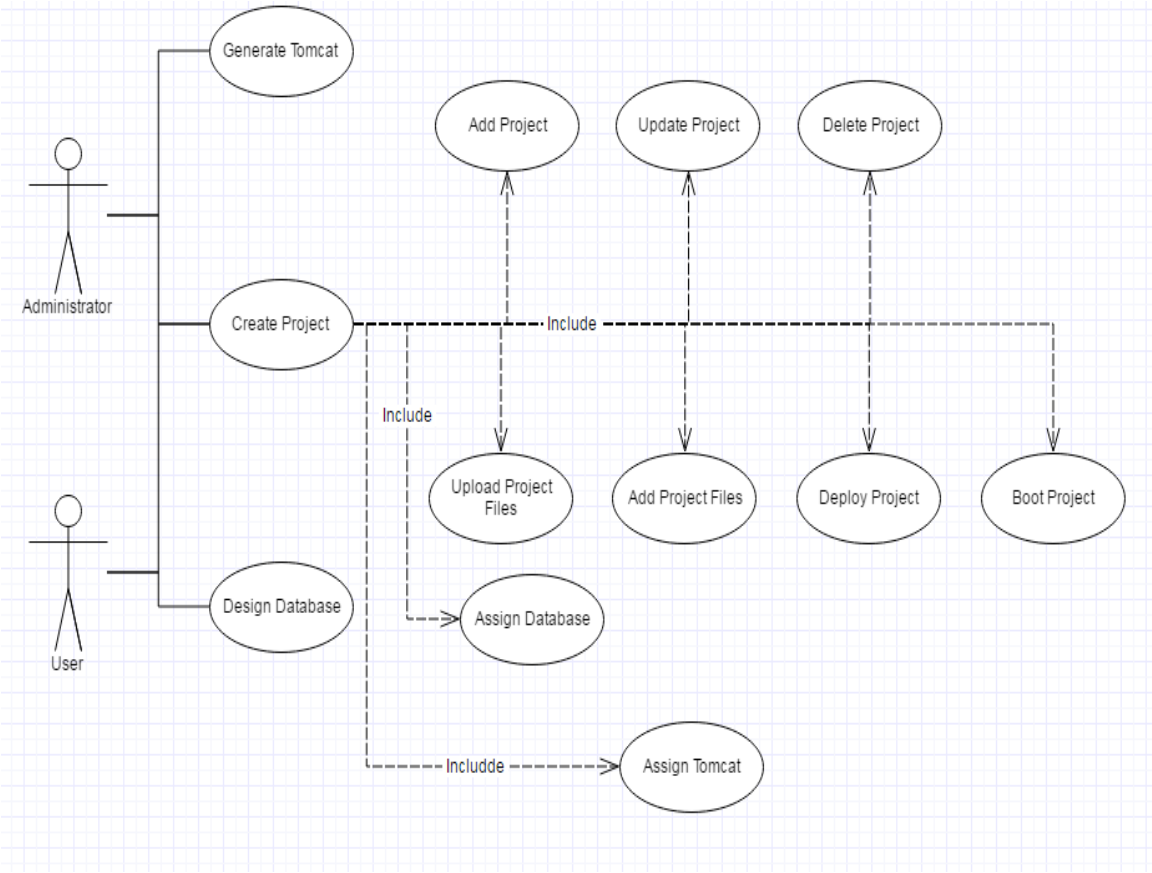



Figure 4.6. Project component use case.

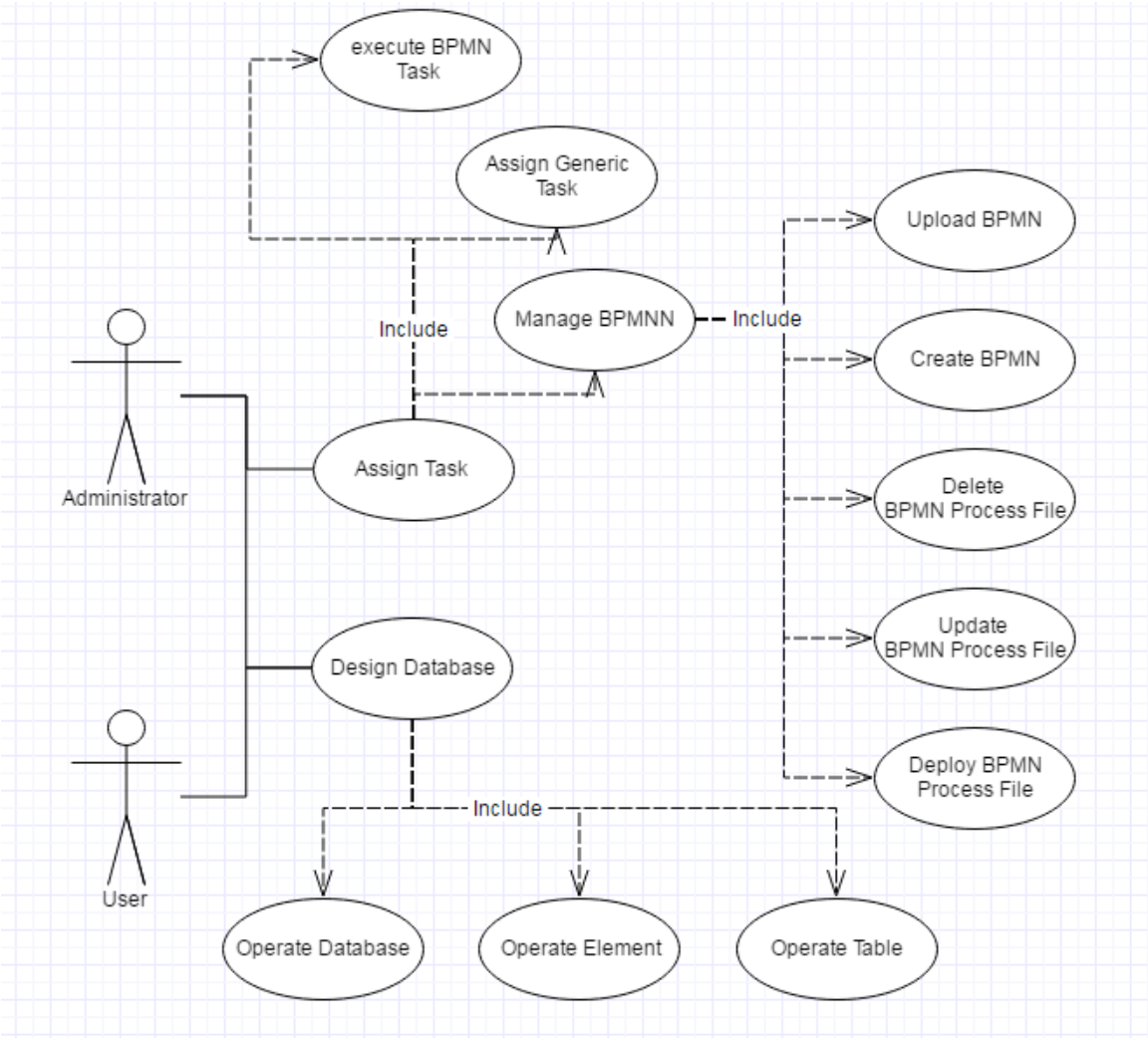



Figure 4.7. Task and database component use case.

Main ER Diagrams

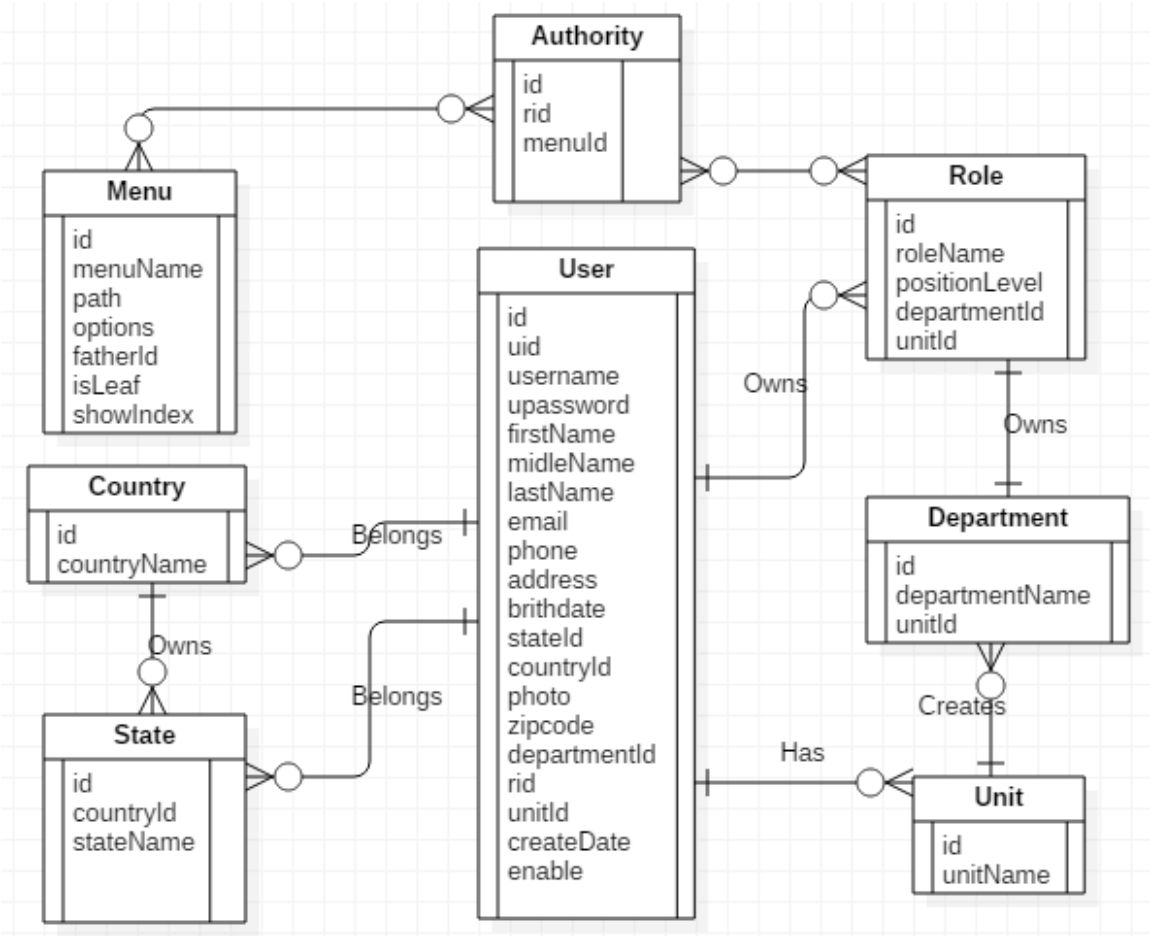



Figure 4.8. Customer ER component.

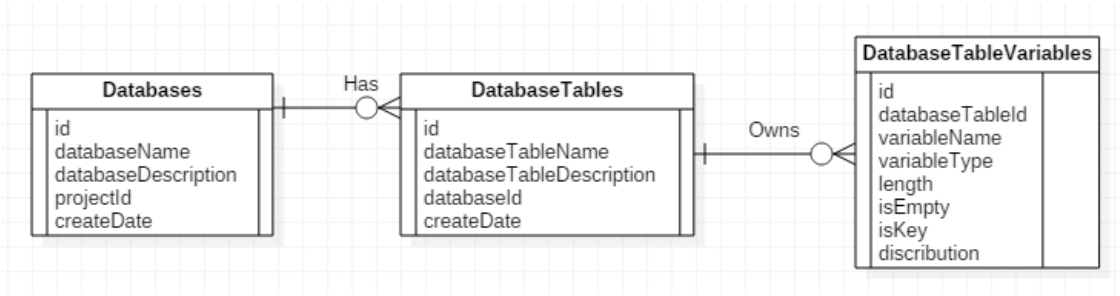



Figure 4.9. Database component.

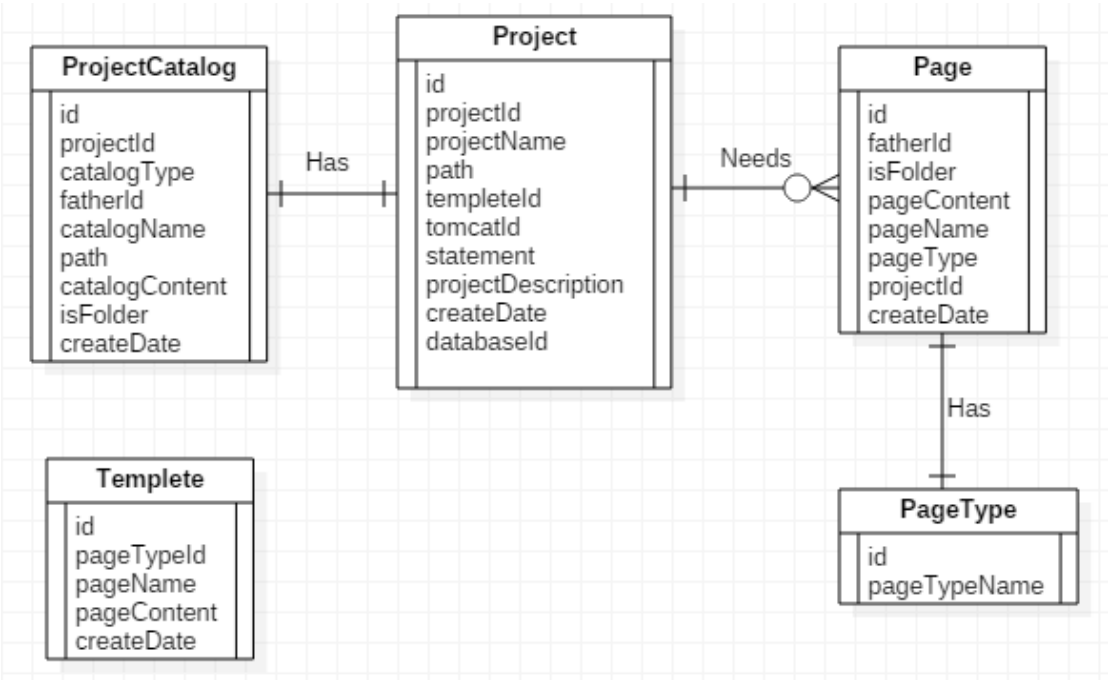



Figure 4.10. Project ER component.

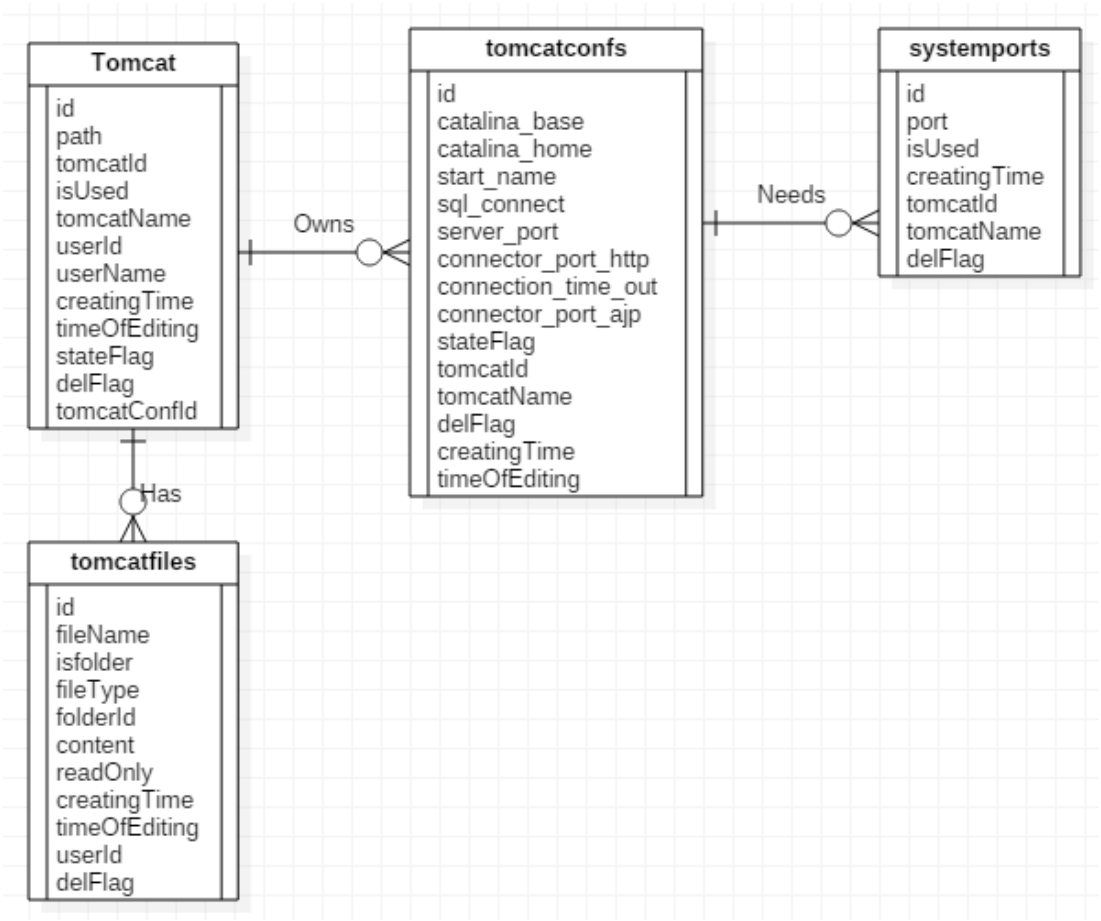
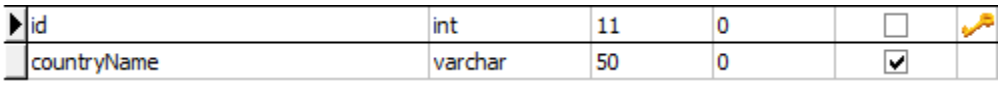



Figure 4.11. Tomcat ER component.

CHAPTER FIVE

PROJECT DATABASE

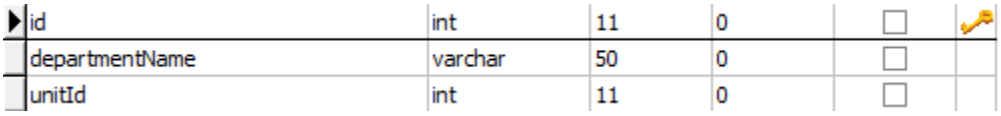


| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| countryName | varchar | 50 | 0 | ☑ | |

Figure 5.1. Table of country.

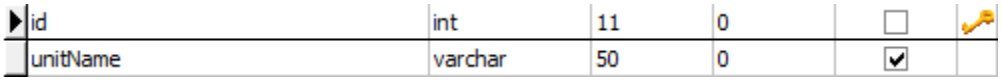
| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| countryId | int | 11 | 0 | ☑ | |
| stateName | varchar | 50 | 0 | ☑ | |

Figure 5.2. Table of state.



| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| departmentName | varchar | 50 | 0 | ☐ | |
| unitId | int | 11 | 0 | ☐ | |

Figure 5.3. Table of department.



| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| unitName | varchar | 50 | 0 | ☑ | |

Figure 5.4. Table of unit.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| rid | int | 11 | 0 | ☐ | |
| menuId | int | 11 | 0 | ☐ | |

Figure 5.5. Table of authorizes.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| variableTypeName | varchar | 50 | 0 | ☐ | |
| variableName | varchar | 255 | 0 | ☑ | |

Figure 5.6. Table of variable.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| roleName | varchar | 50 | 0 | ☐ | |
| positionLevel | int | 11 | 0 | ☐ | |
| departmentId | int | 11 | 0 | ☐ | |
| unitId | int | 11 | 0 | ☐ | |

Figure 5.7. Table of role.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| menuName | varchar | 50 | 0 | ☐ | |
| path | varchar | 50 | 0 | ☑ | |
| options | varchar | 255 | 0 | ☑ | |
| fatherId | int | 11 | 0 | ☐ | |
| isLeaf | int | 11 | 0 | ☐ | |
| showIndex | int | 11 | 0 | ☐ | |

Figure 5.8. Table of menu.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| taskTypeName | varchar | 50 | 0 | ☐ | |

Figure 5.9. Table of task type.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| pageTypeName | varchar | 50 | 0 | ☐ | |

Figure 4.10. Table of page type.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| port | varchar | 255 | 0 | ☐ | |
| isUsed | int | 11 | 0 | ☐ | |
| creatingTime | varchar | 255 | 0 | ☐ | |
| tomcatId | varchar | 255 | 0 | ☑ | |
| tomcatName | varchar | 255 | 0 | ☑ | |
| delFlag | int | 11 | 0 | ☐ | |

Figure 5.11. Table of system port.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| originalTomcatPath | varchar | 100 | 0 | ☑ | |
| targetTomcatPath | varchar | 100 | 0 | ☑ | |
| projectPath | varchar | 100 | 0 | ☑ | |
| photoPath | varchar | 100 | 0 | ☑ | |
| bpmnProcessFilePath | varchar | 100 | 0 | ☑ | |
| pageSize | int | 11 | 0 | ☑ | |

Figure 5.12. Table of system variable configuration.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| uid | varchar | 50 | 0 | ☐ | |
| username | varchar | 50 | 0 | ☐ | |
| upassword | varchar | 50 | 0 | ☐ | |
| firstName | varchar | 50 | 0 | ☑ | |
| midleName | varchar | 50 | 0 | ☑ | |
| lastName | varchar | 50 | 0 | ☑ | |
| email | varchar | 50 | 0 | ☐ | |
| phone | varchar | 50 | 0 | ☑ | |
| address | varchar | 100 | 0 | ☑ | |
| brithdate | varchar | 50 | 0 | ☑ | |
| stateId | int | 11 | 0 | ☑ | |
| countryId | int | 11 | 0 | ☑ | |
| photo | varchar | 100 | 0 | ☑ | |
| zipcode | varchar | 50 | 0 | ☑ | |
| departmentId | int | 50 | 0 | ☑ | |
| rid | int | 11 | 0 | ☐ | |
| unitId | int | 11 | 0 | ☐ | |
| createDate | varchar | 50 | 0 | ☐ | |
| enable | int | 11 | 0 | ☐ | |

Figure 5.13. Table of customer.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| pageTypeId | int | 11 | 0 | ☐ | |
| pageName | varchar | 50 | 0 | ☐ | |
| pageContent | text | 0 | 0 | ☑ | |
| createDate | varchar | 50 | 0 | ☐ | |

Figure 5.14. Table of page type.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| taskName | varchar | 50 | 0 | ☐ | |
| receiverId | varchar | 50 | 0 | ☐ | |
| departmentId | int | 11 | 0 | ☐ | |
| taskFromId | varchar | 50 | 0 | ☐ | |
| taskTypeId | int | 11 | 0 | ☐ | |
| taskContent | text | 0 | 0 | ☐ | |
| isAccept | int | 11 | 0 | ☑ | |
| startDate | varchar | 50 | 0 | ☑ | |
| dueDate | varchar | 50 | 0 | ☐ | |
| process | int | 11 | 0 | ☑ | |
| createDate | varchar | 255 | 0 | ☐ | |

Figure 5.15. Table of task.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| projectId | varchar | 50 | 0 | ☑ | |
| projectName | varchar | 50 | 0 | ☐ | |
| path | varchar | 255 | 0 | ☐ | |
| templeteId | int | 11 | 0 | ☑ | |
| tomcatId | varchar | 50 | 0 | ☑ | |
| statement | int | 11 | 0 | ☑ | |
| projectDescription | text | 0 | 0 | ☑ | |
| createDate | varchar | 50 | 0 | ☐ | |
| databaseId | int | 11 | 0 | ☑ | |

Figure 5.16. Table of project.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| projectId | int | 11 | 0 | ☐ | |
| catalogType | varchar | 50 | 0 | ☐ | |
| fatherId | int | 11 | 0 | ☐ | |
| catalogName | varchar | 50 | 0 | ☐ | |
| path | varchar | 255 | 0 | ☑ | |
| catalogContent | longtext | 0 | 0 | ☑ | |
| isFolder | int | 11 | 0 | ☐ | |
| createDate | varchar | 50 | 0 | ☐ | |

Figure 5.17. Table of project file.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| accountName | varchar | 50 | 0 | ☑ | |
| address | varchar | 100 | 0 | ☑ | |
| cpassword | varchar | 50 | 0 | ☑ | |
| countryId | int | 11 | 0 | ☐ | |
| createDate | varchar | 50 | 0 | ☑ | |
| customerId | varchar | 50 | 0 | ☑ | |
| DOB | varchar | 50 | 0 | ☑ | |
| email | varchar | 50 | 0 | ☑ | |
| enable | int | 11 | 0 | ☐ | |
| firstName | varchar | 50 | 0 | ☑ | |
| midleName | varchar | 50 | 0 | ☑ | |
| lastName | varchar | 50 | 0 | ☑ | |
| phone | varchar | 50 | 0 | ☑ | |
| stateId | int | 11 | 0 | ☐ | |
| zipcode | varchar | 50 | 0 | ☑ | |
| apassword | varchar | 255 | 0 | ☑ | |
| dateofbrith | varchar | 255 | 0 | ☑ | |

Figure 5.18. Table of customer.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| bpmnProcessName | varchar | 50 | 0 | ☐ | |
| realName | varchar | 50 | 0 | ☐ | |
| bpmnProcessPath | varchar | 100 | 0 | ☐ | |
| bpmnDescription | text | 0 | 0 | ☑ | |
| uploadDate | varchar | 50 | 0 | ☐ | |

Figure 5.19. Table of upload BPMN file.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| databaseName | varchar | 50 | 0 | ☐ | |
| databaseDescription | text | 0 | 0 | ☐ | |
| projectId | int | 11 | 0 | ☑ | |
| createDate | varchar | 50 | 0 | ☐ | |

Figure 5.20. Table of database.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| databaseTableName | varchar | 255 | 0 | ☐ | |
| databaseTableDescription | text | 0 | 0 | ☑ | |
| databaseId | int | 11 | 0 | ☐ | |
| createDate | varchar | 255 | 0 | ☐ | |

Figure 5.21. Table of database table.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| databaseTableId | int | 11 | 0 | ☐ | |
| variableName | varchar | 50 | 0 | ☐ | |
| variableType | varchar | 50 | 0 | ☐ | |
| length | int | 11 | 0 | ☐ | |
| isEmpty | int | 11 | 0 | ☐ | |
| isKey | int | 11 | 0 | ☐ | |
| discribution | text | 0 | 0 | ☑ | |

Figure 5.22. Table of database table variable.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| createDate | varchar | 255 | 0 | ☑ | |
| fatherId | int | 11 | 0 | ☐ | |
| isFolder | int | 11 | 0 | ☐ | |
| pageContent | varchar | 255 | 0 | ☑ | |
| pageName | varchar | 255 | 0 | ☑ | |
| pageType | varchar | 255 | 0 | ☑ | |
| projectId | int | 11 | 0 | ☐ | |

Figure 5.23. Table of page.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| title | varchar | 50 | 0 | ☐ | |
| senderName | varchar | 50 | 0 | ☑ | |
| senderId | varchar | 50 | 0 | ☑ | |
| receiverName | varchar | 50 | 0 | ☑ | |
| receiverId | varchar | 50 | 0 | ☑ | |
| content | text | 0 | 0 | ☑ | |
| isRead | int | 11 | 0 | ☐ | |
| isReply | int | 50 | 0 | ☐ | |
| replyId | int | 50 | 0 | ☐ | |
| createDate | varchar | 50 | 0 | ☐ | |

Figure 5.24. Table of message.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| catalina_base | varchar | 255 | 0 | ☐ | |
| catalina_home | varchar | 255 | 0 | ☐ | |
| start_name | varchar | 255 | 0 | ☐ | |
| sql_connect | text | 0 | 0 | ☑ | |
| server_port | varchar | 255 | 0 | ☐ | |
| connector_port_http | varchar | 255 | 0 | ☐ | |
| connection_time_out | varchar | 255 | 0 | ☐ | |
| connector_port_ajp | varchar | 255 | 0 | ☐ | |
| stateFlag | int | 11 | 0 | ☐ | |
| tomcatId | varchar | 255 | 0 | ☐ | |
| tomcatName | varchar | 255 | 0 | ☐ | |
| delFlag | int | 11 | 0 | ☐ | |
| creatingTime | varchar | 255 | 0 | ☐ | |
| timeOfEditing | varchar | 255 | 0 | ☐ | |

Figure 5.25. Table of tomcat config.

| | | | | | |
|---|---|---|---|---|---|
| id | varchar | 255 | 0 | ☐ | 🔑 |
| fileName | varchar | 50 | 0 | ☐ | |
| isfolder | int | 11 | 0 | ☐ | |
| fileType | int | 11 | 0 | ☑ | |
| folderId | varchar | 255 | 0 | ☐ | |
| content | text | 0 | 0 | ☑ | |
| readOnly | int | 11 | 0 | ☐ | |
| creatingTime | varchar | 50 | 0 | ☐ | |
| timeOfEditing | varchar | 50 | 0 | ☐ | |
| userId | int | 11 | 0 | ☐ | |
| delFlag | int | 11 | 0 | ☐ | |

Figure 5.26. Table of file.

| | | | | | |
|---|---|---|---|---|---|
| id | int | 11 | 0 | ☐ | 🔑 |
| path | varchar | 255 | 0 | ☐ | |
| tomcatId | varchar | 255 | 0 | ☐ | |
| isUsed | int | 11 | 0 | ☐ | |
| tomcatName | varchar | 255 | 0 | ☐ | |
| userId | int | 11 | 0 | ☐ | |
| userName | varchar | 50 | 0 | ☐ | |
| creatingTime | varchar | 50 | 0 | ☐ | |
| timeOfEditing | varchar | 50 | 0 | ☐ | |
| stateFlag | int | 11 | 0 | ☐ | |
| delFlag | int | 11 | 0 | ☐ | |
| tomcatConfId | int | 11 | 0 | ☑ | |

Figure 5.27. Table of tomcat.

# CHAPTER SIX

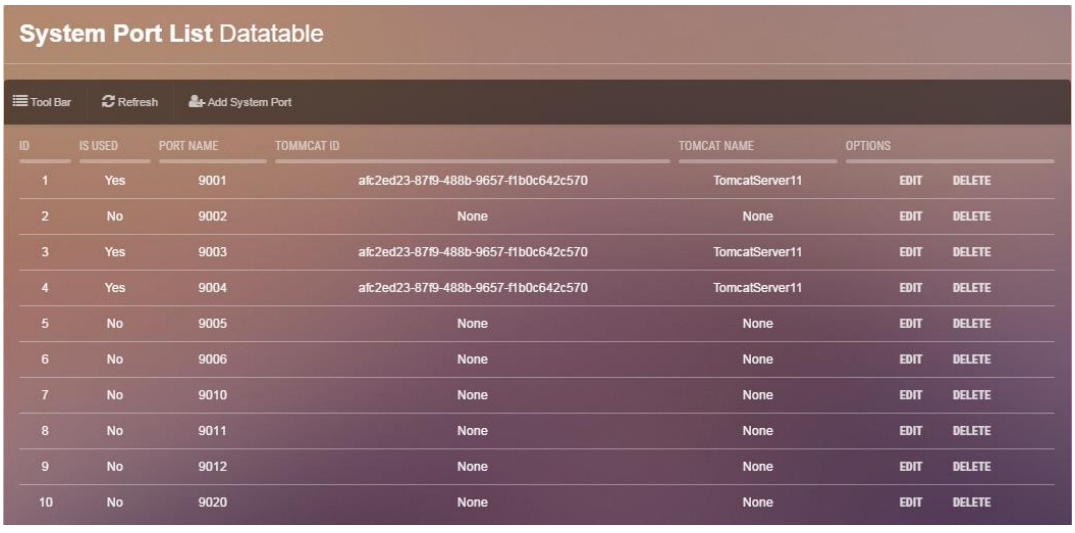
# PROJECT COMPONENT

## System Configuration

When login into the system, the system administrator needs to setup serval items as Figure 6.1 shows. The first item is the root folder path of original tomcat and target tomcat folder path where customer tomcat should be saved at. The original tomcat folder has been put with the project together deployed to tomcat service project folder. The second item is to setup the project path where customer's project should be stored into. The third item is to setup photo path where all upload pictures should be saved. The fourth item is BPMN file path where all upload BPMN process files go to and where the BPMN engine should look for deployed process files. The last one is page size that can determine how many items should show on a list page.


Figure 6.1. System variable configuration.

The system port configuration (Figure 6.2) is for tomcat web service. Every tomcat web service needs three system ports so that the system administrator needs to check and make sure that there has enough system ports for the customer to use.  Otherwise, the customers cannot create tomcat web service.




| ID | IS USED | PORT NAME | TOMMCAT ID | TOMCAT NAME | OPTIONS | |
|---|---|---|---|---|---|---|
| 1 | Yes | 9001 | afc2ed23-87f9-488b-9657-f1b0c642c570 | TomcatServer11 | EDIT | DELETE |
| 2 | No | 9002 | None | None | EDIT | DELETE |
| 3 | Yes | 9003 | afc2ed23-87f9-488b-9657-f1b0c642c570 | TomcatServer11 | EDIT | DELETE |
| 4 | Yes | 9004 | afc2ed23-87f9-488b-9657-f1b0c642c570 | TomcatServer11 | EDIT | DELETE |
| 5 | No | 9005 | None | None | EDIT | DELETE |
| 6 | No | 9006 | None | None | EDIT | DELETE |
| 7 | No | 9010 | None | None | EDIT | DELETE |
| 8 | No | 9011 | None | None | EDIT | DELETE |
| 9 | No | 9012 | None | None | EDIT | DELETE |
| 10 | No | 9020 | None | None | EDIT | DELETE |

Figure 6.2. System port configuration.

The variable type (Figure 6.3) is for creating databases. Customers can create databases through this project and they need these variables type to declare every element.


Figure 6.3. Variable type configuration.

The page type configuration (Figure 6.4) is for creating web pages. Customers can create project file trees with different web file types and folders such as JSP, js, or CSS. If the file is a JSP, then customers can write those files in JSP language. After customers deploy their projects, all file will be deployed to their tomcat web services as well.


Figure 6.4. Page type configuration.

The task type configuration (Figure 6.5) is for BPMN task. The customers can assign different type of BPMN tasks to other customers.


Figure 6.5. Task type configuration.

Setup Customer Model

To set up a company account, the system administrator needs to use country and state configuration (Figure 6.6 and 3.7).


Figure 6.6. State configuration.

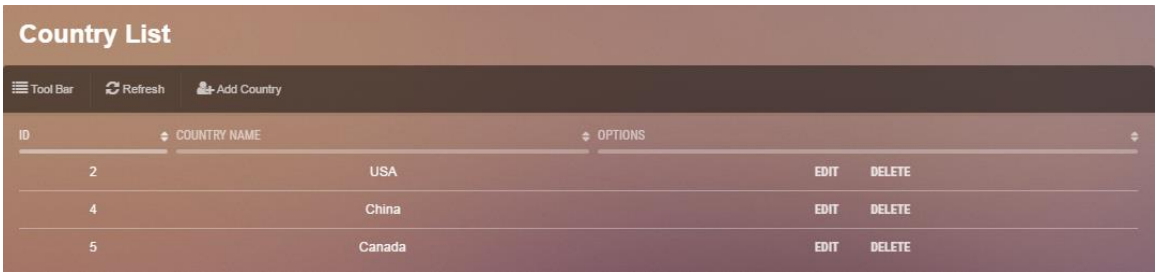



Figure 6.7. Country configuration.

The company configuration (Figure 6.8) is used to separate different companies. When enterprise customers sign in the system, they will not see each other.


Figure 6.8. Company configuration.

As we know, every company has many departments. Every department has its own roles. Under this system, The number of menus customers can access are controlled by the role configuration. Every company has a system level administrator who can access the role (Figure 6.9), department (Figure 6.10), and menu (Figure 6.11) tab, but the administrator can only allocate the rights he/she has.


Figure 6.9. Role configuration.


Figure 6.10. Department configuration.

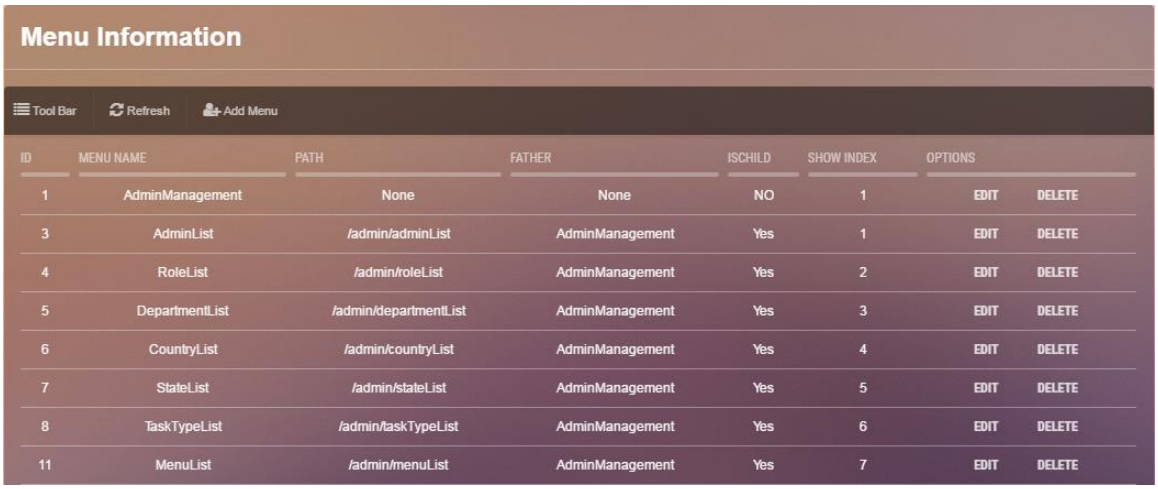



Figure 6.11. Country configuration.

## Setup Tomcat Web Service

To add tomcat services, the customers need to go to the project management tab. Under that tab, the customers can find tomcat web service list (Figure 6.12). To add tomcats, the customers need to click add tomcat button at the toolbar of a display list, then the customers will see add tomcat page (Figure 6.). Now, they need to input a unique tomcat web service name, because the system will check if the name is a unique name.

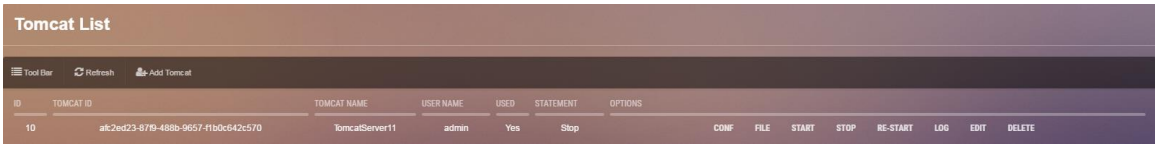



Figure 6.12. Tomcat web service list.


Figure 6.13. Add tomcat web service page.

When the customers add their tomcat web service successfully, the customers are able to get into their tomcat fie configuration page (Figure 6.14). The customers then will see the tomcat configuration list. On the list, the customers can add many configuration files and can only choose one of them to apply to the tomcat. After then, customers will see very basic information of the tomcat configuration (Figure 6.15), including server port, HTTP port, ALP port, and current tomcat configuration status.

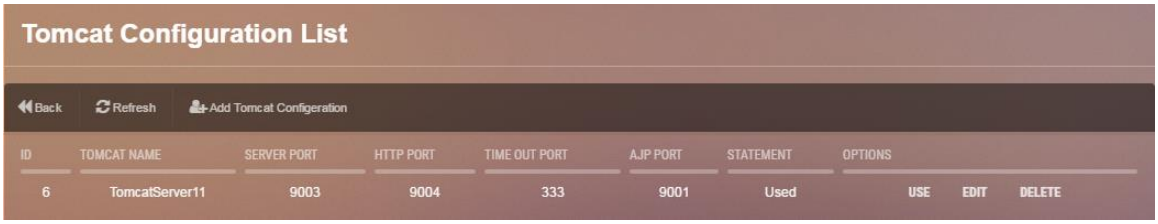



Figure 6.14. Tomcat configuration list.

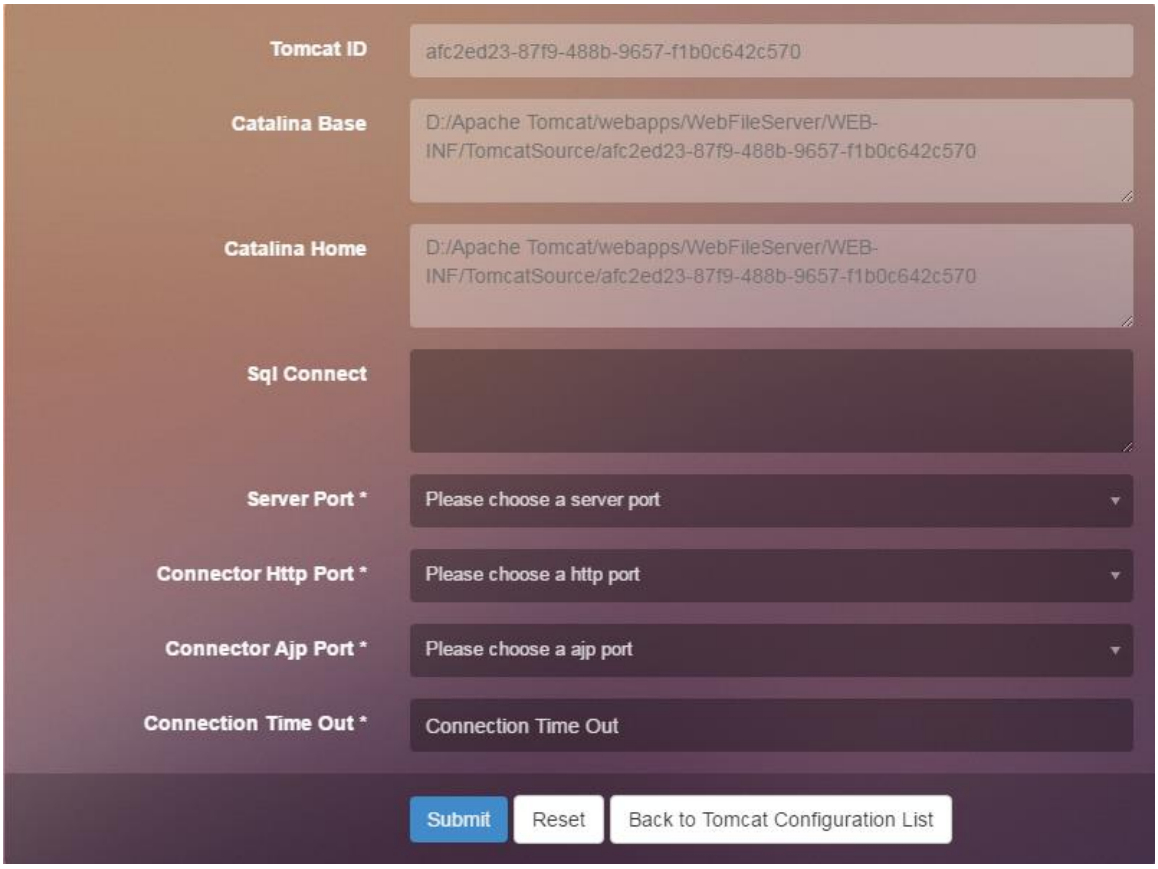



Figure 6.15. Add tomcat web service page.

After the customer tomcat servers are created, the customers can see the tomcat file tree at tomcat file list page (Figure 6.16). At there, the customers can also see a physical file as Figure 6.17 shows.


Figure 6.16. Tomcat configuration list.

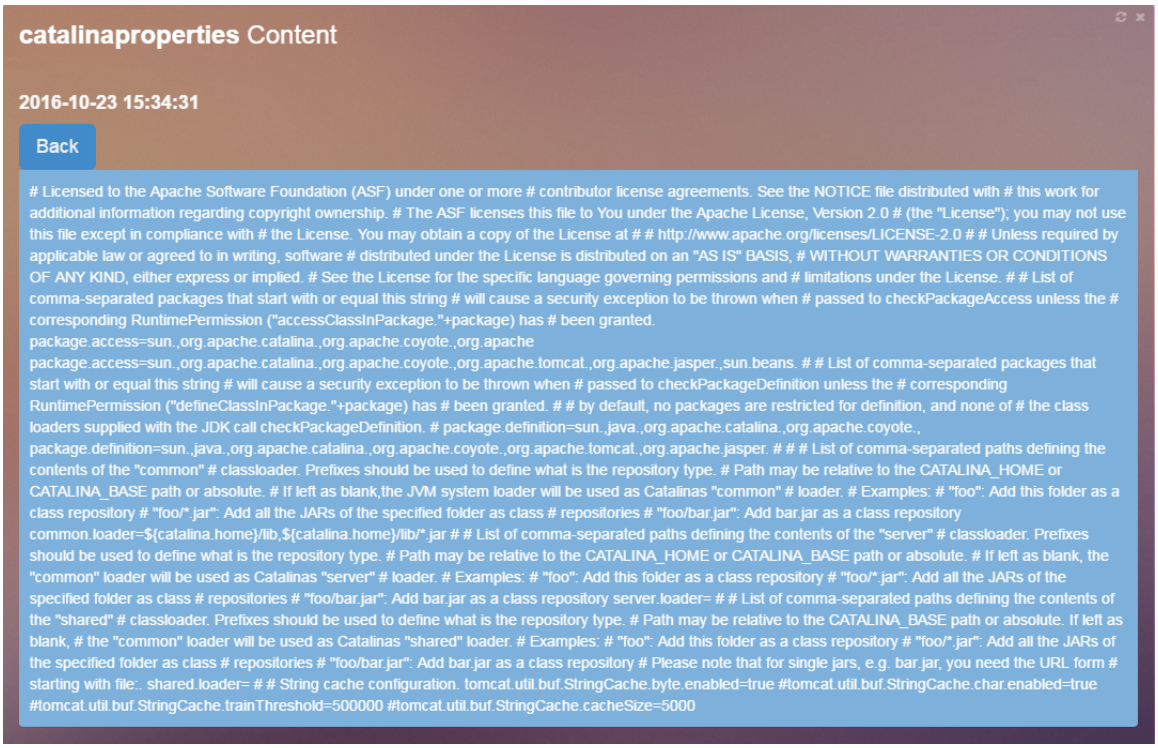



Figure 6.17. Tomcat file content.

The customers can customize the tomcat file and folder as Figure 6.18 and 3.19 shows. The customers can also add an extra tomcat running jar into tomcat lib for their projects.

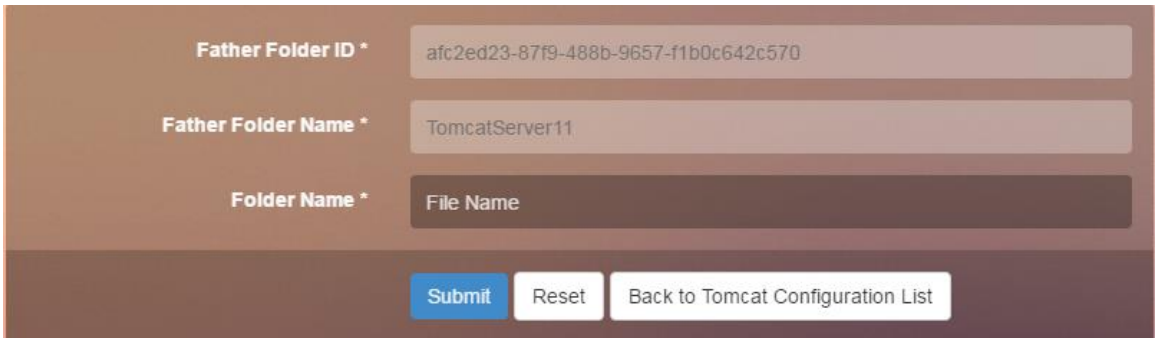



Figure 6.18. Add tomcat folder.


Figure 6.19. Add tomcat file.

When everything has been setup, the customer can do these tomcat operations that are start, stop, and re-start at the tomcat service list page.

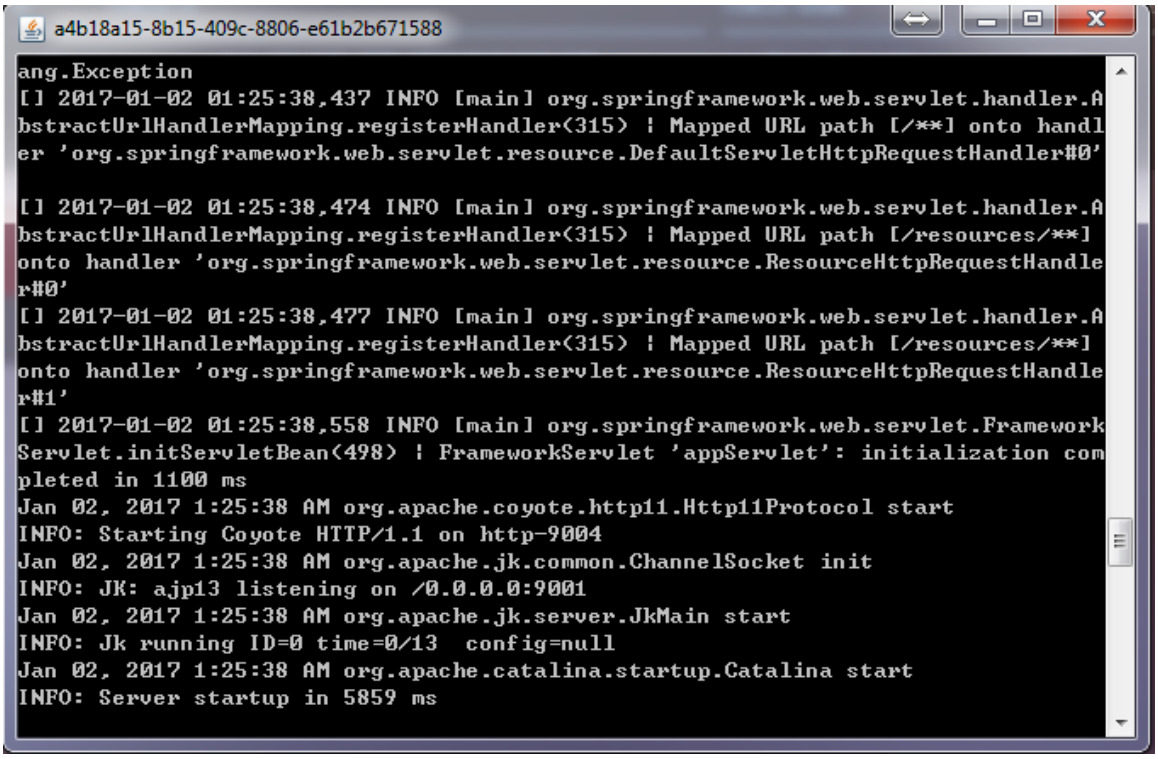



Figure 6.19. Tomcat web service console.

Creating Project

Under the project management, the customers can add projects to the project list (Figure 20) after the customers create tomcat web services.


Figure 6.20. Project list.

To add a project, customers need to input at least the project name and project description what the customer will write some note to explain what the project will do and what it is.

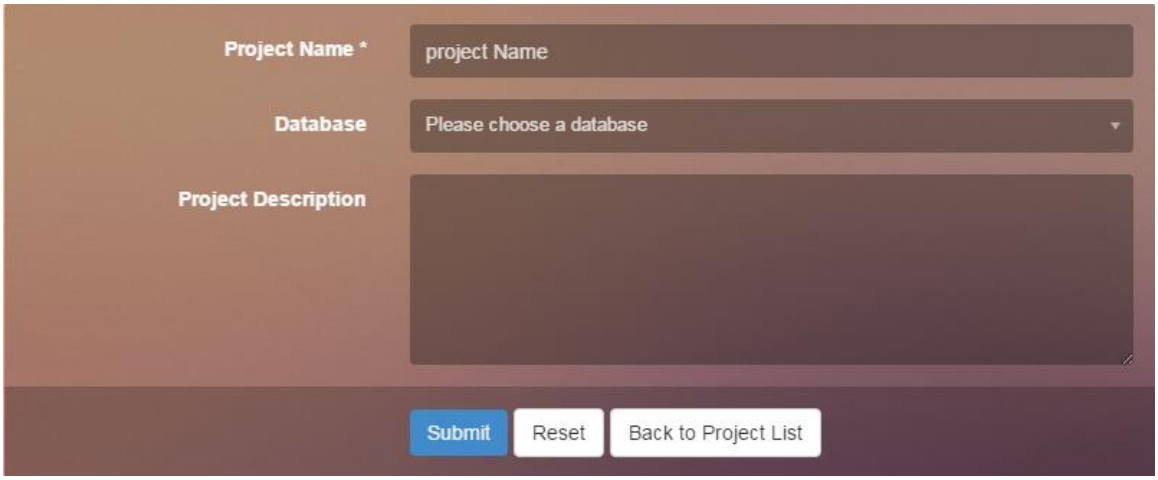



Figure 6.21. Add project page.

When the project has been created, the customer can determine what type of the project file he/she wants to do. If the customer chooses to create the project file with clicking add catalog button (Figure 22), the customer will create an online coding file. If the customer chooses to click the upload catalog button (Figure 22), the customer will upload a tomcat project war file to the tomcat web service and run that project.

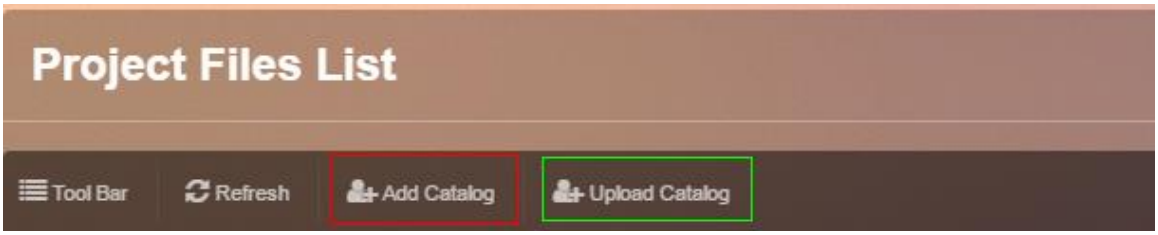



Figure 6.22 Two ways to add project files.

Deploying Project and Start Web Service

When the customer has his/her project file done, the customer needs to boot the project first. Boot the project (Figure 6.24) means project files the customer created online will be written to disk from database and then the customer needs to click upload button (Figure 6.24) to upload these files to tomcat web service. Then, the customer can start his/her tomcat web service successfully.

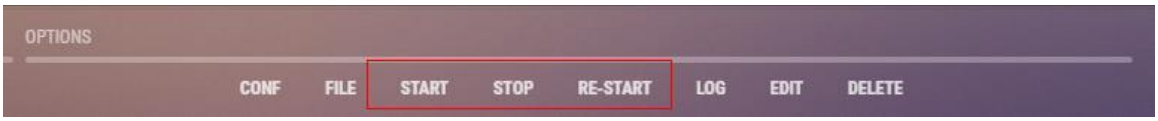



Figure 6.23. Tomcat web service options.

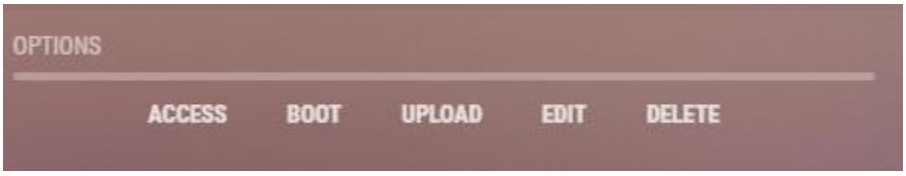



Figure 6.24. Project options.

Deploying BPMN Process Flow

To deploy a BPMN process file, a customer needs to determine which way he/she wants to use. I provide two ways to deploy BPMN process file. Under the project management tab, there have two BPMN display lists. The first list (Figure 6.25) will show a customer internal BPMN file list what all BPMN process files are created by the system with embedded BPMN file editor (Figure 6.30). Another list (Figure 6.26) is for the customer to upload and deploy a BPMN process file to BPMN engine. The external BPMN will be shown at external BPMN display list (Figure 6.27).

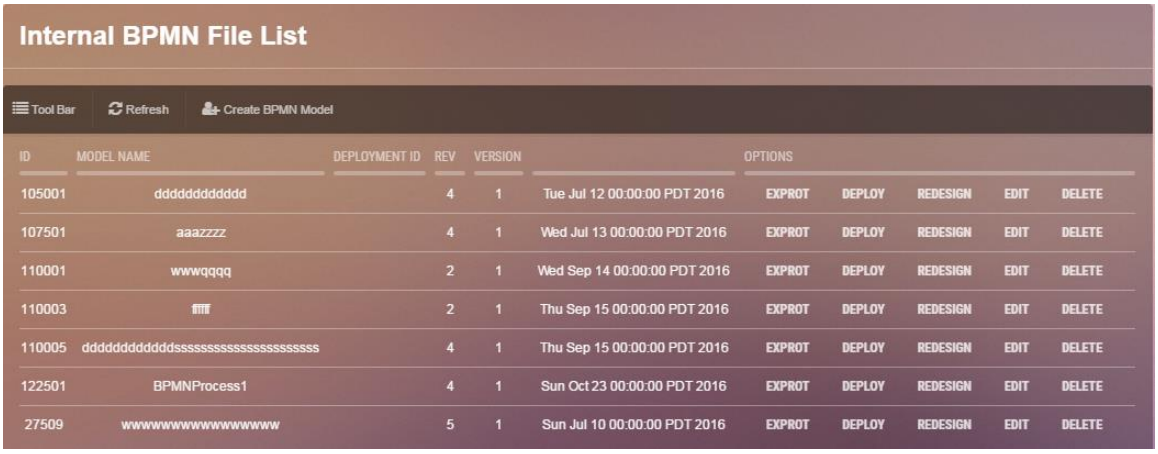



Figure 6.25. Created on system side BPMN process list.

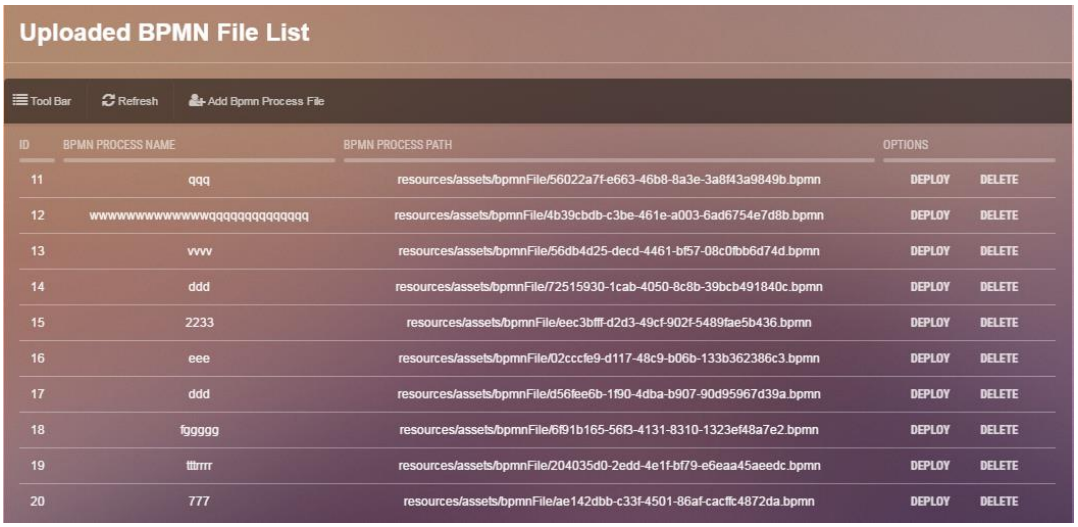



Figure 6.26. Uploaded BPMN process file list.

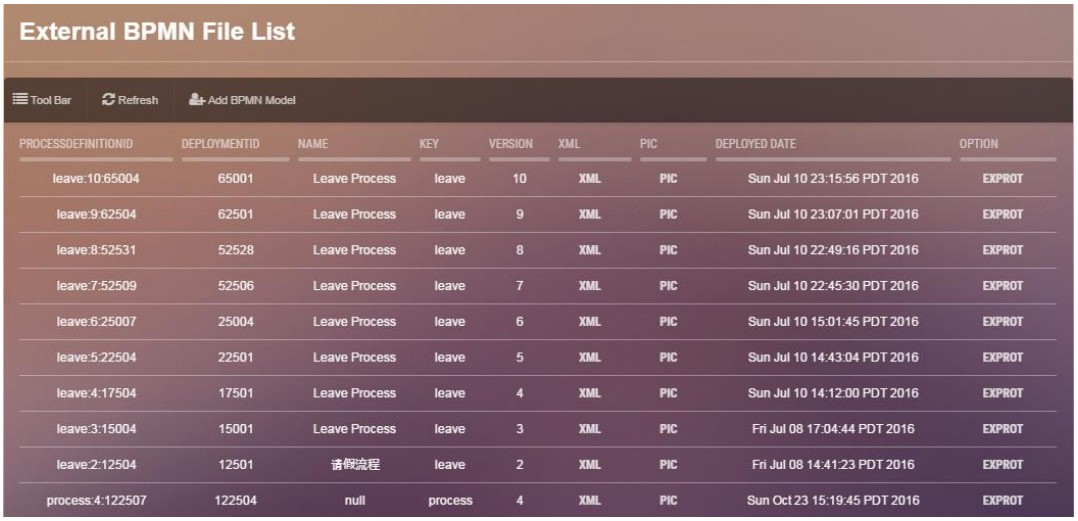



Figure 6.27. Deployed uploaded BPMN process list.

At external BPMN display list, the customer can review a BPMN process file in two ways. The first is to see the process file in XML format (Figure 6.28). Another way is to see the process file as an image (Figure 6.29).

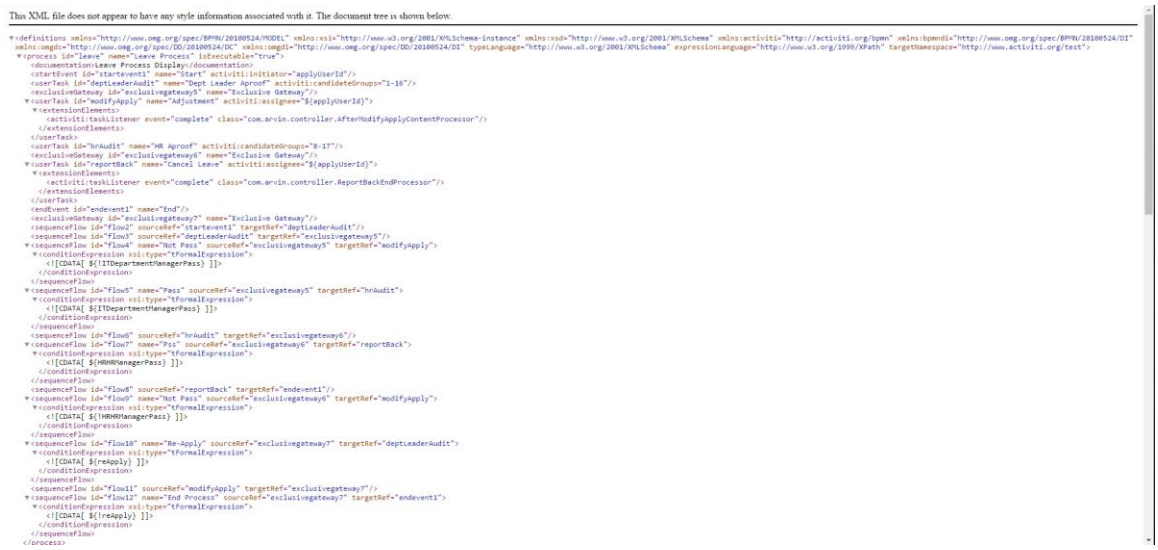


Figure 6.28. BPMNN process file in xml format.

This BPMN leave process displays if customers request leaves, the BPMN engine will start the processes and pass the task to department leader. If the department leader accepts these requests, the engine will pass this task to HR department. If the HR department leader also accept these request, then the requests will be sent back customers. If the either HR department leader or department leader does not accept these reqirements, the reqirements will also be sent back to customers, but customers can either reject these reqirements or do some adjustments to restart these reqirement processes.

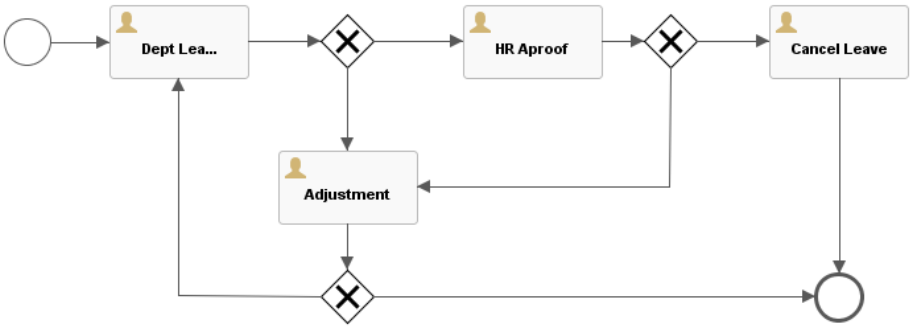



Figure 6.29. BPMNN process displayed by image.

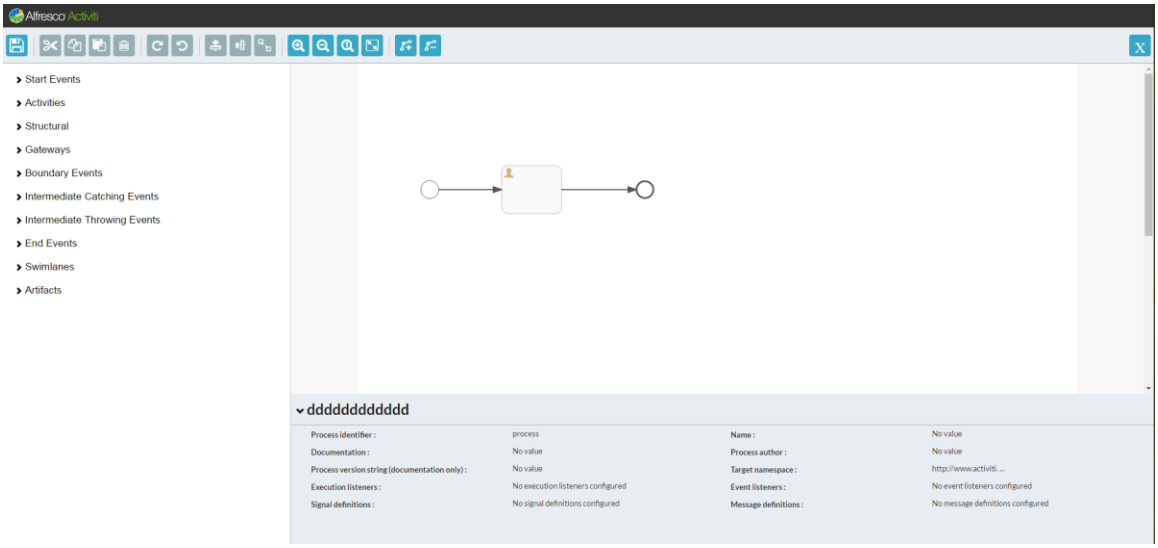



Figure 6.30. Embedded BPMN process editor.

## Assign Tasks

This system provides task component which a customer can assign a task to other customers or he/she can also request a task like leave requirement. I design the task component contains two types of task. the first type is BPMN task (Figure 6.31) and another one is the generic task (Figure 6.35 and 3. 36). The BPMN task page has three data lists that they are the current task (Figure 6.31), in process task (Figure 6.32), and finished the task (Figure 6.33).


Figure 6.31. Current BPMN process task list.

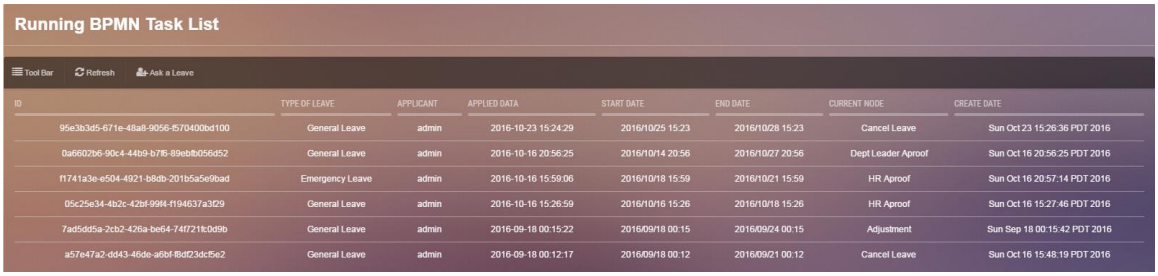



Figure 6.32. In process BPMN task list.

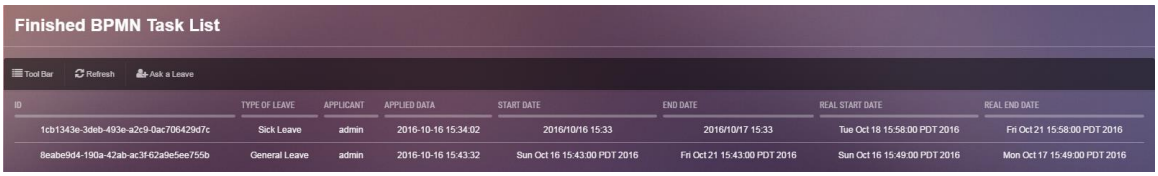



Figure 6.33. Finished BPMN task list.

To add a BPMN task, a customer needs to fill out a BPMN task form (Figure 6.34). The form includes some basic information that will help department leader and HR department leader to determine if he/she needs to reject the request or accept it.


Figure 6.34. Add BPMN task page.

To assign generic tasks (Figure 6.35), customers should have high-level roles such as department leader. After customers assign generic tasks to other customers. The receivers can find the task at either system tool (Figure 6.36) bar, received task list page (Figure 6.37), or to-do list (Figure 6.38).

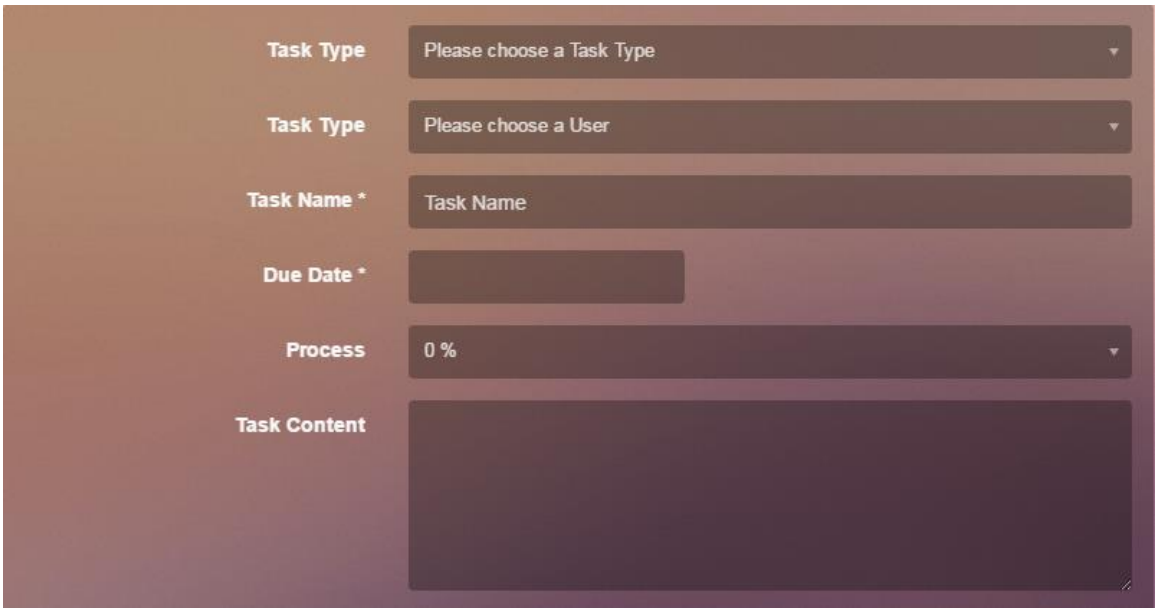



Figure 6.35. Add generic task page.


Figure 6.36. Received task list.

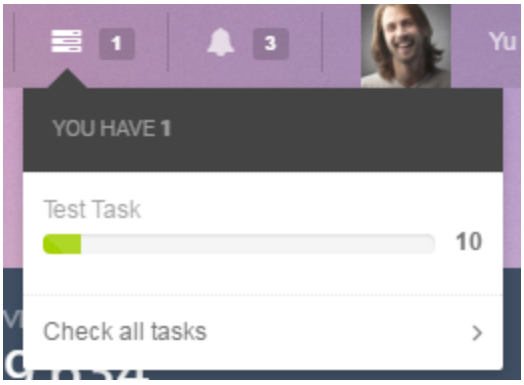



Figure 6.37. System tool bar task list.

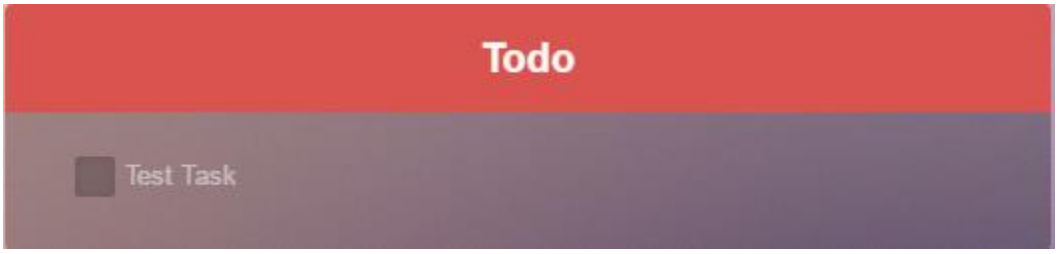



Figure 6.38. To do list of task.

Once a customer finishes a generic task. he/she can click that task to setup the task as completed as Figure 6.39 shows.

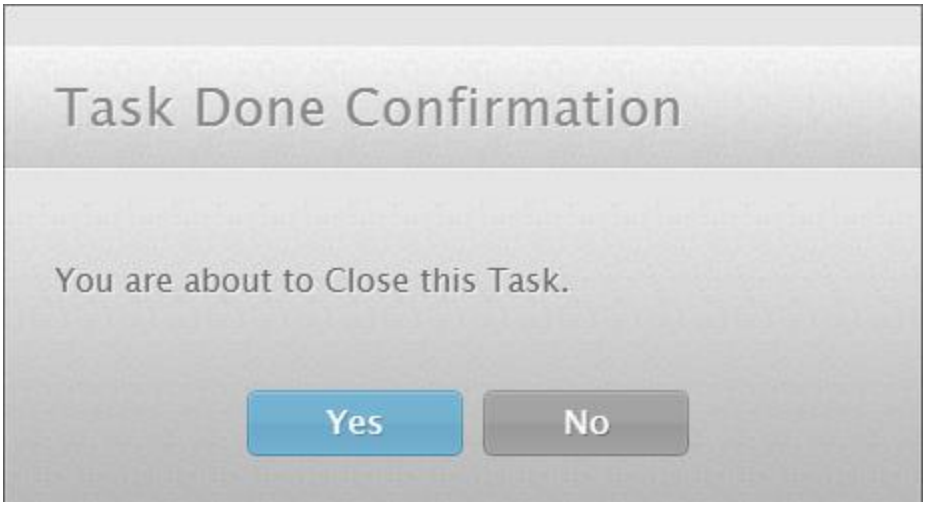



Figure 6.39. Finish a task.

Send Message

The message component allow customers communicate easier with each other by sending an internal message. Then, the receivers are able to find the message at message list page (Figure 6.40) or system toolbar (Figure 6.41).

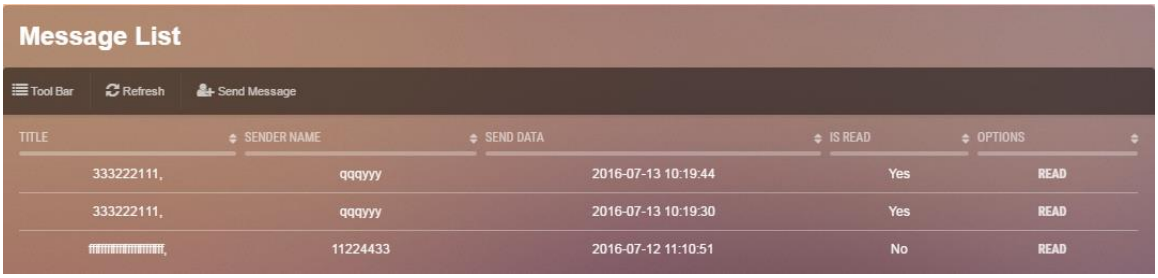



Figure 6.40. Message list.

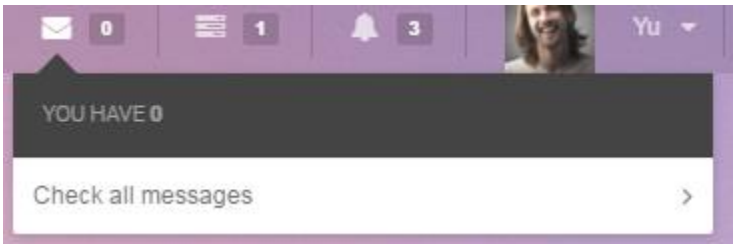



Figure 6.41. System tool bar message list.

# CHAPTER SEVEN

# CONCLUSION

In the web application generator market of North America, Weebly, WordPress, and Website Builder are widely used web application generators, but these web application generators still cannot generate 100% suitable web application projects for their customers. All the web application generators ask their customers to choose a web template first and then the customers need to modify or add more data to the template at their online editor page. Due to those web application generator, there is no such website page tree diagram which can allow the customers to have an overview of their website constructions. Based on these points, the customers need to know at least how to design and layout their website pages. After the customers have done some adjustments for their websites, the templates can then fit to their business. The project can generate web applications by clicking one button according to the BPMN (Business Process Model Notation) technology. The BPMN engine will become a middle layer that will lead to the logic of page jumping. The customers can modify the content of their web applications through the administration platform with a website page tree diagram. The customers can also manage their web application data at the data section.

This project provides both generic and customized methods to generate web applications. When using the generic method to generate web applications, the customers need to go to the main page and choose one of the existing web

application models and then fill out the template data. By using the customized way to design web applications, the customers should go to the special order page and fill out extra work order forms for the web application support team. Based on the special orders, the support team will pass those work orders to development department. The development department will give suitable development methodologies such as waterfall, agile, and prototype to fit with the customer requirements and develop these requirements. Based on a selected development methodology, customers will receive their required product after one software development life cycle has been implemented.

# APPENDIX

# CODE OF CRITICAL PARTS

```
<developers>
	<developer>
	  <id>Arvin</id>
	  <name>Yu Zhou</name>
	  <email>zhouyuzero@hotmail.com</email>
	  <timezone>8</timezone>
	</developer>
</developers>

<properties>
	<java-version>1.8</java-version>
	<jetty.version>8.1.15.v20140411</jetty.version>
	<org.springframework-version>4.0.3.RELEASE</org.springframework-
version>
	<org.aspectj-version>1.7.4</org.aspectj-version>
	<org.slf4j-version>1.7.5</org.slf4j-version>
	<hibernate.version>4.3.5.Final</hibernate.version>
	<jackson.version>2.6.3</jackson.version>
	<activiti.version>5.19.0</activiti.version>
	<guava.version>17.0</guava.version>
	<commons-lang3.version>3.3.2</commons-lang3.version>
	<joda-time.version>2.1</joda-time.version>
	<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<repositories>
		<repository>
			<id>Activiti</id>

	  <url>https://maven.alfresco.com/nexus/content/groups/public</url>
		</repository>
		<repository>
			<id>Maven Central</id>
			<url>http://repo.maven.apache.org/maven2/</url>
		</repository>
		<repository>
	  <id>nexus-osc</id>
	  <url>http://maven.oschina.net/content/groups/public/</url>
	</repository>
	<repository>
	  <id>nexus-osc-thirdparty</id>
	  <url>http://maven.oschina.net/content/repositories/thirdparty/</url>
```

```
        </repository>
</repositories>

<build>
                <finalName>${project.artifactId}</finalName>
                <resources>
          <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
          </resource>
        </resources>
                <plugins>
                        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
              <source>${java-version}</source>
              <target>${java-version}</target>
              <showWarnings>true</showWarnings>
            </configuration>
          </plugin>
          <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
               <server>apache-tomcat-7</server>
              <path>/${project.artifactId}</path>
              <url>http://localhost:8080/onlinewebsitegenerator</url>
              <systemProperties>
                 <JAVA_OPTS>-Xms1024m -Xmx2048m -XX:PermSize=512m -
XX:MaxPermSize=1024m -Dfile.encoding=UTF-8</JAVA_OPTS>
              </systemProperties>
            </configuration>
          </plugin>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.4</version>
          </plugin>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-eclipse-plugin</artifactId>
```

```
            <version>2.9</version>
            <configuration>
               <wtpversion>2.0</wtpversion>
               <additionalProjectnatures>

<projectnature>org.springframework.ide.eclipse.core.springnature</projectnature
>
               </additionalProjectnatures>
               <additionalBuildcommands>

<buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcomm
and>
               </additionalBuildcommands>
               <downloadSources>true</downloadSources>
               <downloadJavadocs>true</downloadJavadocs>
            </configuration>
         </plugin>
                     <plugin>
            <groupId>org.mortbay.jetty</groupId>
            <artifactId>jetty-maven-plugin</artifactId>
            <version>${jetty.version}</version>
            <configuration>
               <webApp>
                  <contextPath>/${project.artifactId}</contextPath>
               </webApp>
               <stopKey>${project.artifactId}</stopKey>
               <stopPort>9999</stopPort>
            </configuration>
         </plugin>
                     <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-resources-plugin</artifactId>
            <version>2.6</version>
            <configuration>
               <nonFilteredFileExtensions>
                  <nonFilteredFileExtension>zip</nonFilteredFileExtension>
                  <nonFilteredFileExtension>bar</nonFilteredFileExtension>
                  <nonFilteredFileExtension>png</nonFilteredFileExtension>
                  <nonFilteredFileExtension>bpmn</nonFilteredFileExtension>
               </nonFilteredFileExtensions>
               <encoding>${project.build.sourceEncoding}</encoding>
            </configuration>
         </plugin>
          </plugins>
```

```
</build>
```

Web Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">

        <display-name>Online Website Generator Application</display-name>

        <listener>
                <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
        <listener>
                <listener-
class>org.springframework.web.util.Log4jConfigListener</listener-class>
        </listener>
        <listener>
                <listener-
class>com.arvin.sessionlistener.SessionListener</listener-class>
        </listener>

        <session-config>
                <session-timeout>30</session-timeout>
        </session-config>

        <context-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>
        /WEB-INF/spring-dao.xml,
        /WEB-INF/spring-service.xml,
        /WEB-INF/spring-bpmn.xml
      </param-value>
        </context-param>

        <servlet>
                <servlet-name>appServlet</servlet-name>
                <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
                <init-param>
                        <param-name>contextConfigLocation</param-name>
                        <param-value>/WEB-INF/spring-servlet.xml</param-value>
                </init-param>
```

```
            <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
            <servlet-name>appServlet</servlet-name>
            <url-pattern>/</url-pattern>
        </servlet-mapping>
        <!--
        <servlet>
            <servlet-name>RestletServlet</servlet-name>
        <servlet>
            <servlet-name>ModelRestServlet</servlet-name>
            <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
            <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring-modeler.xml</param-
value>
            </init-param>
            <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
            <servlet-name>ModelRestServlet</servlet-name>
            <url-pattern>/service/*</url-pattern>
        </servlet-mapping>
        <servlet>
            <servlet-name>RestServlet</servlet-name>
            <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
            <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring-rest.xml</param-value>
            </init-param>
            <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
            <servlet-name>RestServlet</servlet-name>
            <url-pattern>/rest/*</url-pattern>
        </servlet-mapping>

        <context-param>
            <param-name>log4jConfigLocation</param-name>
            <param-value>/WEB-INF/log4j.properties</param-value>
        </context-param>
        <context-param>
```

```
        <param-name>log4jRefreshInterval</param-name>
        <param-value>60000</param-value>
    </context-param>

    <filter>
        <filter-name>CharacterEncodingFilter</filter-name>
        <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>
            org.springframework.web.filter.DelegatingFilterProxy
        </filter-class>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/rest/*</url-pattern>
        <dispatcher>ERROR</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>

    <filter>
        <filter-name>SessionFilter</filter-name>
        <filter-class>com.arvin.sessionlistener.SessionFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>SessionFilter</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>REQUEST</dispatcher>
```

```
        </filter-mapping>

        <error-page>
                <error-code>404</error-code>
                <location>/WEB-INF/pages/page404.jsp</location>
        </error-page>
        <error-page>
                <error-code>500</error-code>
                <location>/WEB-INF/pages/page500.jsp</location>
        </error-page>
</web-app>
```

Spring Configuration

Servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
          http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        ">
    <annotation-driven/>
    <default-servlet-handler />
    <context:component-scan base-package="com.arvin.controller" />

    <resources location="/resources/" mapping="/resources/**" />
    <resources location="/editor/" mapping="/editor/**"/>
          <resources location="/explorer/" mapping="/explorer/**"/>
          <resources location="/libs/" mapping="/libs/**"/>
          <resources location="/api/" mapping="/api/**"/>
          <resources location="/editor-app/" mapping="/editor-app/**"/>
          <resources location="/diagram-viewer/" mapping="/diagram-viewer/**"/>
          <resources location="/images/" mapping="/images/**"/>

    <beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/pages/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <beans:bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver
">
                    <!-- setting maximum upload size -->
                  <beans:property name="maxUploadSize" value="50000000" />
          </beans:bean>
```

```
        <beans:bean id="mappingJacksonHttpMessageConverter"
class="org.springframework.http.converter.json.MappingJackson2HttpMessageC
onverter">
      <beans:property name="supportedMediaTypes">
        <beans:list>
          <beans:value>text/html;charset=UTF-8</beans:value>
        </beans:list>
      </beans:property>
   </beans:bean>

   <beans:bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandl
erAdapter">
      <beans:property name="messageConverters">
        <beans:list>
          <beans:ref bean="mappingJacksonHttpMessageConverter" />
        </beans:list>
      </beans:property>
   </beans:bean>

   <!-- Configuring interceptors based on URI -->
       <interceptors>
      <interceptor>
        <mapping path="/*" />
        <beans:bean
class="com.arvin.interceptor.RequestProcessingTimeInterceptor">
          <beans:property name="excludedUrls">
            <beans:list>
              <beans:value>/</beans:value>
              <beans:value>/customerLogin</beans:value>
              <beans:value>/upload</beans:value>
              <beans:value>/timeout</beans:value>
            </beans:list>
          </beans:property>
        </beans:bean>
      </interceptor>
   </interceptors>

        <beans:bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandl
erAdapter">
     <beans:property name="messageConverters">
       <beans:list>
```

```
            <beans:bean class =
"org.springframework.http.converter.StringHttpMessageConverter">
                <beans:property name = "supportedMediaTypes">
                    <beans:list>
                        <beans:value>text/html;charset=UTF-8</beans:value>
                    </beans:list>
                </beans:property>
            </beans:bean>
        </beans:list>
    </beans:property>
</beans:bean>

    <beans:bean id="serviceManager"
class="com.arvin.service.manager.ServiceManager">
        <!-- **********Admin********** -->
        <beans:property name="customersService">
            <beans:ref bean="customersService" />
        </beans:property>

        <!-- **********Menu********** -->
        <beans:property name="rolesService">
            <beans:ref bean="rolesService" />
        </beans:property>
        <beans:property name="menusService">
            <beans:ref bean="menusService" />
        </beans:property>

        <!-- **********Task********** -->
        <beans:property name="tasksService">
            <beans:ref bean="tasksService" />
        </beans:property>

        <!-- **********Message********** -->
        <beans:property name="messagesService">
            <beans:ref bean="messagesService" />
        </beans:property>

        <!-- **********System Variables********** -->
        <beans:property name="stateService">
            <beans:ref bean="stateService" />
        </beans:property>
        <beans:property name="countryService">
            <beans:ref bean="countryService" />
        </beans:property>
```

```
<beans:property name="departmentsService">
	<beans:ref bean="departmentsService" />
</beans:property>
<beans:property name="taskTypesService">
	<beans:ref bean="taskTypesService" />
</beans:property>
<beans:property name="pageTypesService">
	<beans:ref bean="pageTypesService" />
</beans:property>
<beans:property name="unitsService">
	<beans:ref bean="unitsService" />
</beans:property>
<beans:property name="systemPortsService">
	<beans:ref bean="systemPortsService" />
</beans:property>
<beans:property name="systemVariablesService">
	<beans:ref bean="systemVariablesService" />
</beans:property>
<beans:property name="variableTypesService">
	<beans:ref bean="variableTypesService" />
</beans:property>

<!-- **********Project********** -->
<beans:property name="projectsService">
	<beans:ref bean="projectsService" />
</beans:property>

<!-- **********Database********** -->
<beans:property name="databasesService">
	<beans:ref bean="databasesService" />
</beans:property>
<beans:property name="databaseTablesService">
	<beans:ref bean="databaseTablesService" />
</beans:property>
<beans:property name="databaseTableVariablesService">
	<beans:ref bean="databaseTableVariablesService" />
</beans:property>

<!-- **********Tomcat********** -->
<beans:property name="tomcatsService">
	<beans:ref bean="tomcatsService" />
</beans:property>
<beans:property name="tomcatFilesService">
	<beans:ref bean="tomcatFilesService" />
```

```
        </beans:property>
        <beans:property name="tomcatConfsService">
            <beans:ref bean="tomcatConfsService" />
        </beans:property>

        <!-- **********************BPMN************************ -->
        <beans:property name="act_re_model_Service">
            <beans:ref bean="act_re_model_Service" />
        </beans:property>
        <beans:property name="act_ge_bytearray_Service">
            <beans:ref bean="act_ge_bytearray_Service" />
        </beans:property>
        <beans:property name="leavesService">
            <beans:ref bean="leavesService" />
        </beans:property>
        <beans:property name="bpmnService">
            <beans:ref bean="bpmnService" />
        </beans:property>
        <beans:property name="act_id_group_Service">
            <beans:ref bean="act_id_group_Service" />
        </beans:property>
        <beans:property name="act_id_customer_Service">
            <beans:ref bean="act_id_customer_Service" />
        </beans:property>

        <!-- ************************ProjectCatalogs************************** -->
        <beans:property name="projectCatalogsService">
            <beans:ref bean="projectCatalogsService" />
        </beans:property>
</beans:bean>
</beans:beans>
```

Service

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:beans="http://www.springframework.org/schema/beans"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- **********Admin********** -->
        <beans:bean id="customersService"
class="com.arvin.service.impl.CustomersServiceImpl">
                <beans:property name="customersDao"
ref="customersDao"></beans:property>
        </beans:bean>

        <!-- **********Menu********** -->
        <beans:bean id="menusService"
class="com.arvin.service.impl.MenusServiceImpl">
                <beans:property name="menusDao"
ref="menusDao"></beans:property>
        </beans:bean>
        <beans:bean id="rolesService"
class="com.arvin.service.impl.RolesServiceImpl">
                <beans:property name="rolesDao"
ref="rolesDao"></beans:property>
        </beans:bean>

        <!-- **********Task********** -->
        <beans:bean id="tasksService"
class="com.arvin.service.impl.TasksServiceImpl">
                <beans:property name="tasksDao"
ref="tasksDao"></beans:property>
        </beans:bean>

        <!-- **********Message********** -->
        <beans:bean id="messagesService"
class="com.arvin.service.impl.MessagesServiceImpl">
                <beans:property name="messagesDao"
ref="messagesDao"></beans:property>
        </beans:bean>

        <!-- **********System Variables********** -->
```

```
        <beans:bean id="stateService"
class="com.arvin.service.impl.StateServiceImpl">
                <beans:property name="stateDao"
ref="stateDao"></beans:property>
        </beans:bean>
        <beans:bean id="countryService"
class="com.arvin.service.impl.CountryServiceImpl">
                <beans:property name="countryDao"
ref="countryDao"></beans:property>
        </beans:bean>
        <beans:bean id="departmentsService"
class="com.arvin.service.impl.DepartmentsServiceImpl">
                <beans:property name="departmentsDao"
ref="departmentsDao"></beans:property>
        </beans:bean>
        <beans:bean id="taskTypesService"
class="com.arvin.service.impl.TaskTypesServiceImpl">
                <beans:property name="taskTypesDao"
ref="taskTypesDao"></beans:property>
        </beans:bean>
        <beans:bean id="unitsService"
class="com.arvin.service.impl.UnitsServiceImpl">
                <beans:property name="unitsDao"
ref="unitsDao"></beans:property>
        </beans:bean>
        <beans:bean id="pageTypesService"
class="com.arvin.service.impl.PageTypesServiceImpl">
                <beans:property name="pageTypesDao"
ref="pageTypesDao"></beans:property>
        </beans:bean>
        <beans:bean id="systemPortsService"
class="com.arvin.service.impl.SystemPortsServiceImpl">
                <beans:property name="systemPortsDao"
ref="systemPortsDao"></beans:property>
        </beans:bean>
        <beans:bean id="systemVariablesService"
class="com.arvin.service.impl.SystemVariablesServiceImpl">
                <beans:property name="systemVariablesDao"
ref="systemVariablesDao"></beans:property>
        </beans:bean>
        <beans:bean id="variableTypesService"
class="com.arvin.service.impl.VariableTypesServiceImpl">
                <beans:property name="variableTypesDao"
ref="variableTypesDao"></beans:property>
```

```
        </beans:bean>

        <!-- **********Project********** -->
        <beans:bean id="projectsService"
class="com.arvin.service.impl.ProjectsServiceImpl">
                <beans:property name="projectsDao"
ref="projectsDao"></beans:property>
        </beans:bean>

        <!-- **********Tomcat********** -->
        <beans:bean id="databasesService"
class="com.arvin.service.impl.DatabasesServiceImpl">
                <beans:property name="databasesDao"
ref="databasesDao"></beans:property>
        </beans:bean>
        <beans:bean id="databaseTablesService"
class="com.arvin.service.impl.DatabaseTablesServiceImpl">
                <beans:property name="databaseTablesDao"
ref="databaseTablesDao"></beans:property>
        </beans:bean>
        <beans:bean id="databaseTableVariablesService"
class="com.arvin.service.impl.DatabaseTableVariablesServiceImpl">
                <beans:property name="databaseTableVariablesDao"
ref="databaseTableVariablesDao"></beans:property>
        </beans:bean>

        <!-- **********Database********** -->
        <beans:bean id="tomcatsService"
class="com.arvin.service.impl.TomcatsServiceImpl">
                <beans:property name="tomcatsDao"
ref="tomcatsDao"></beans:property>
        </beans:bean>
        <beans:bean id="tomcatFilesService"
class="com.arvin.service.impl.TomcatFilesServiceImpl">
                <beans:property name="tomcatFilesDao"
ref="tomcatFilesDao"></beans:property>
        </beans:bean>
        <beans:bean id="tomcatConfsService"
class="com.arvin.service.impl.TomcatConfsServiceImpl">
                <beans:property name="tomcatConfsDao"
ref="tomcatConfsDao"></beans:property>
        </beans:bean>

        <!-- **********BPMN********** -->
```

```
		<beans:bean id="act_re_model_Service"
class="com.arvin.service.impl.Act_re_model_Service_Impl">
			<beans:property name="act_re_model_Dao"
ref="act_re_model_Dao"></beans:property>
		</beans:bean>
		<beans:bean id="act_ge_bytearray_Service"
class="com.arvin.service.impl.Act_ge_bytearray_Service_Impl">
			<beans:property name="act_ge_bytearray_Dao"
ref="act_ge_bytearray_Dao"></beans:property>
		</beans:bean>
		<beans:bean id="leavesService"
class="com.arvin.service.impl.LeavesServiceImpl">
			<beans:property name="leavesDao"
ref="leavesDao"></beans:property>
		</beans:bean>
		<beans:bean id="bpmnService"
class="com.arvin.service.impl.BpmnServiceImpl">
			<beans:property name="bpmnDao"
ref="bpmnDao"></beans:property>
		</beans:bean>
		<beans:bean id="act_id_customer_Service"
class="com.arvin.service.impl.Act_id_customer_Service_Impl">
			<beans:property name="act_id_customer_Dao"
ref="act_id_customer_Dao"></beans:property>
		</beans:bean>
		<beans:bean id="act_id_group_Service"
class="com.arvin.service.impl.Act_id_group_Service_Impl">
			<beans:property name="act_id_group_Dao"
ref="act_id_group_Dao"></beans:property>
		</beans:bean>

		<!-- **********ProjectCatalogs********** -->
		<beans:bean id="projectCatalogsService"
class="com.arvin.service.impl.ProjectCatalogsServiceImpl">
			<beans:property name="projectCatalogsDao"
ref="projectCatalogsDao"></beans:property>
		</beans:bean>
		</beans>
```

BPMN Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:context="http://www.springframework.org/schema/context"
            xsi:schemaLocation="
            http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context.xsd">

   <context:component-scan base-
package="org.activiti.conf,org.activiti.rest.editor,org.activiti.rest.service">
      <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
   </context:component-scan>

         <bean id="objectMapper"
class="com.fasterxml.jackson.databind.ObjectMapper"/>

         <bean id="uuidGenerator"
class="org.activiti.engine.impl.persistence.StrongUuidGenerator" />

   <bean id="processEngine"
class="org.activiti.spring.ProcessEngineFactoryBean">
                  <property name="processEngineConfiguration"
ref="processEngineConfiguration" />
         </bean>

   <bean id="processEngineConfiguration"
class="org.activiti.spring.SpringProcessEngineConfiguration">
                  <property name="dataSource" ref="dataSource" />
      <property name="databaseSchemaUpdate" value="true" />
      <property name="jobExecutorActivate" value="false"/>
      <!--<property name="history" value="full"/>-->
      <property name="transactionManager" ref="transactionManager" />
      <property name="processDefinitionCacheLimit" value="10"/>
        </bean>

         <bean id="repositoryService" factory-bean="processEngine" factory-
method="getRepositoryService" />
         <bean id="runtimeService" factory-bean="processEngine" factory-
method="getRuntimeService" />
```

```
        <bean id="taskService" factory-bean="processEngine" factory-
method="getTaskService" />
        <bean id="historyService" factory-bean="processEngine" factory-
method="getHistoryService" />
        <bean id="managementService" factory-bean="processEngine" factory-
method="getManagementService" />
    <bean id="IdentityService" factory-bean="processEngine" factory-
method="getIdentityService" />
    <bean id="formService" factory-bean="processEngine" factory-
method="getFormService" />

    <bean id="restResponseFactory"
class="org.activiti.rest.service.api.RestResponseFactory" />
    <bean id="contentTypeResolver"
class="org.activiti.rest.common.application.DefaultContentTypeResolver" />
        </beans>
```

Hibernate Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
        ">
        <tx:annotation-driven transaction-manager="transactionManager" proxy-
target-class="true"/>

        <beans:bean id="propertyConfigurer"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
>
        <beans:property name="location" value="/WEB-
INF/database.properties"/>
        </beans:bean>

        <beans:bean id="dataSource"
class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
                <!-- Connection Info -->
            <beans:property name="driverClassName"
value="${jdbc.driverClassName}" />
       <beans:property name="url" value="${jdbc.url}" />
       <beans:property name="customername" value="${jdbc.customername}" />
       <beans:property name="password" value="${jdbc.password}" />
       <!-- Connection Pooling Info -->
       <beans:property name="maxActive" value="${dbcp.maxActive}"/>
       <beans:property name="maxIdle" value="${dbcp.maxIdle}"/>
       <!-- Idle -->
       <beans:property name="defaultAutoCommit" value="false"/>
       <beans:property name="timeBetweenEvictionRunsMillis"
value="3600000"/>
       <beans:property name="minEvictableIdleTimeMillis" value="3600000"/>
   </beans:bean>

   <beans:bean id="transactionManager"
class="org.springframework.orm.hibernate4.HibernateTransactionManager">
       <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
```

```
    </beans:bean>

    <beans:bean id="hibernate4AnnotatedSessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <beans:property name="dataSource" ref="dataSource" />
        <beans:property name="annotatedClasses">
            <beans:list>
                <beans:value>com.arvin.entity.Customers</beans:value>
                <beans:value>com.arvin.entity.Roles</beans:value>
                <beans:value>com.arvin.entity.Customers</beans:value>
                <beans:value>com.arvin.entity.State</beans:value>
                <beans:value>com.arvin.entity.Country</beans:value>
                <beans:value>com.arvin.entity.Tasks</beans:value>
                <beans:value>com.arvin.entity.TaskTypes</beans:value>
                <beans:value>com.arvin.entity.Messages</beans:value>
                <beans:value>com.arvin.entity.Departments</beans:value>
                <beans:value>com.arvin.entity.Menus</beans:value>
                <beans:value>com.arvin.entity.Authorities</beans:value>
                <beans:value>com.arvin.entity.Units</beans:value>
                <beans:value>com.arvin.entity.Projects</beans:value>
                <beans:value>com.arvin.entity.Databases</beans:value>
                <beans:value>com.arvin.entity.DatabaseTables</beans:value>

<beans:value>com.arvin.entity.DatabaseTableVariables</beans:value>
                <beans:value>com.arvin.entity.Templetes</beans:value>
                <beans:value>com.arvin.entity.TomcatConfs</beans:value>
                <beans:value>com.arvin.entity.TomcatFiles</beans:value>
                <beans:value>com.arvin.entity.Tomcats</beans:value>
                <beans:value>com.arvin.entity.VariableTypes</beans:value>
                <beans:value>com.arvin.entity.PageTypes</beans:value>
                <beans:value>com.arvin.entity.SystemVariables</beans:value>
                <beans:value>com.arvin.entity.SystemPorts</beans:value>
                <beans:value>com.arvin.entity.ProjectCatalogs</beans:value>
                <beans:value>com.arvin.entity.activiti.Act_re_model</beans:value>

<beans:value>com.arvin.entity.activiti.Act_ge_bytearray</beans:value>
                <beans:value>com.arvin.entity.BpmnProcessFiles</beans:value>
            </beans:list>
        </beans:property>
        <beans:property name="hibernateProperties">
            <beans:props>
                <beans:prop
key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</beans:prop>
```

```
            <beans:prop
key="hibernate.ejb.naming_strategy">org.hibernate.cfg.ImprovedNamingStrateg
y</beans:prop>
            <beans:prop
key="hibernate.show_sql">${hibernate.show_sql}</beans:prop>
            <beans:prop
key="hibernate.dialect">${hibernate.dialect}</beans:prop>
            <beans:prop key="hibernate.format_sql">true</beans:prop>
            <beans:prop key="hibernate.hbm2ddl.auto">update</beans:prop>
            <beans:prop
key="hibernate.connection.useUnicode">true</beans:prop>
            <beans:prop key="hibernate.connection.characterEncoding">UTF-
8</beans:prop>
            <beans:prop key="hibernate.max_fetch_depth">0</beans:prop>
            <beans:prop
key="hibernate.jdbc.use_scrollable_resultset">true</beans:prop>
            <beans:prop key="hibernate.jdbc.fetch_size">30</beans:prop>
                        <beans:prop
key="hibernate.jdbc.batch_size">20</beans:prop>
                        <beans:prop
key="hibernate.cache.use_second_level_cache">${hibernate.cache.use_second
_level_cache}</beans:prop>
                        <beans:prop
key="hibernate.cache.use_query_cache">${hibernate.cache.use_query_cache}<
/beans:prop>
                        <beans:prop
key="hibernate.cache.region.factory_class">${hibernate.cache.region.factory_cla
ss}</beans:prop>
                        <beans:prop
key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ca
che.provider_configuration_file_resource_path}</beans:prop>
          </beans:props>
        </beans:property>
          </beans:bean>

          <beans:bean id="transactionInterceptor"
class="org.springframework.transaction.interceptor.TransactionInterceptor">
                <beans:property name="transactionManager">
                        <beans:ref bean="transactionManager" />
                </beans:property>
                <beans:property name="transactionAttributes">
                        <beans:props>
                        <beans:prop
key="delete*">PROPAGATION_REQUIRED</beans:prop>
```

```
                        <beans:prop
key="operate*">PROPAGATION_REQUIRED,-Exception</beans:prop>
                        <beans:prop
key="insert*">PROPAGATION_REQUIRED,-Exception</beans:prop>
                        <beans:prop
key="update*">PROPAGATION_REQUIRED,-Exception</beans:prop>
                        <beans:prop
key="save*">PROPAGATION_REQUIRED</beans:prop>
                        <beans:prop
key="find*">PROPAGATION_REQUIRED,readOnly</beans:prop>
                    </beans:props>
                </beans:property>
        </beans:bean>

        <beans:bean id="txProxy"
class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCrea
tor">
                <beans:property name="beanNames">
                    <beans:list>
                        <beans:value>*Service*</beans:value>
                    </beans:list>
                </beans:property>
                <beans:property name="interceptorNames">
                    <beans:list>
                        <beans:value>transactionInterceptor</beans:value>
                    </beans:list>
                </beans:property>
        </beans:bean>

        <!-- **********************Admin************************ -->
        <beans:bean id="customersDao"
class="com.arvin.dao.impl.CustomersDaoImpl">
      <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
   </beans:bean>

   <!-- **********************Message************************ -->
   <beans:bean id="messagesDao"
class="com.arvin.dao.impl.MessagesDaoImpl">
      <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
   </beans:bean>

   <!-- **********************Task************************ -->
```

```
        <beans:bean id="tasksDao" class="com.arvin.dao.impl.TasksDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********Menu********** -->
    <beans:bean id="menusDao" class="com.arvin.dao.impl.MenusDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="rolesDao" class="com.arvin.dao.impl.RolesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********System Variables********** -->
    <beans:bean id="countryDao" class="com.arvin.dao.impl.CountryDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="stateDao" class="com.arvin.dao.impl.StateDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="departmentsDao"
class="com.arvin.dao.impl.DepartmentsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="taskTypesDao"
class="com.arvin.dao.impl.TaskTypesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="unitsDao" class="com.arvin.dao.impl.UnitsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="systemPortsDao"
class="com.arvin.dao.impl.SystemPortsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
```

```
    <beans:bean id="systemVariablesDao"
class="com.arvin.dao.impl.SystemVariablesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="variableTypesDao"
class="com.arvin.dao.impl.VariableTypeDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="pageTypesDao"
class="com.arvin.dao.impl.PageTypesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********Project********** -->
          <beans:bean id="projectsDao"
class="com.arvin.dao.impl.ProjectsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********Database********** -->
          <beans:bean id="databasesDao"
class="com.arvin.dao.impl.DatabasesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
          <beans:bean id="databaseTablesDao"
class="com.arvin.dao.impl.DatabaseTablesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
          <beans:bean id="databaseTableVariablesDao"
class="com.arvin.dao.impl.DatabaseTableVariablesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********Tomcat********** -->
          <beans:bean id="tomcatsDao"
class="com.arvin.dao.impl.TomcatsDaoImpl">
```

```
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
          <beans:bean id="tomcatFilesDao"
class="com.arvin.dao.impl.TomcatFilesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="tomcatConfsDao"
class="com.arvin.dao.impl.TomcatConfsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>

    <!-- **********************BPMN************************ -->
          <beans:bean id="act_re_model_Dao"
class="com.arvin.dao.impl.Act_re_model_Dao_Impl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="act_ge_bytearray_Dao"
class="com.arvin.dao.impl.Act_ge_bytearray_Dao_Impl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="leavesDao" class="com.arvin.dao.impl.LeavesDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="bpmnDao" class="com.arvin.dao.impl.BpmnDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="act_id_group_Dao"
class="com.arvin.dao.impl.Act_id_group_Dao_Impl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
    <beans:bean id="act_id_customer_Dao"
class="com.arvin.dao.impl.Act_id_customer_Dao_Impl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
```

```
    <!-- **********************ProjectCatalogs************************** -->
    <beans:bean id="projectCatalogsDao"
class="com.arvin.dao.impl.ProjectCatalogsDaoImpl">
        <beans:property name="sessionFactory"
ref="hibernate4AnnotatedSessionFactory" />
    </beans:bean>
          </beans>
```

```
package com.arvin.controller;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;

import com.arvin.entity.Country;
import com.arvin.entity.Databases;
import com.arvin.entity.Departments;
import com.arvin.entity.Menus;
import com.arvin.entity.PageTypes;
import com.arvin.entity.Roles;
import com.arvin.entity.State;
import com.arvin.entity.SystemPorts;
import com.arvin.entity.SystemVariables;
import com.arvin.entity.TaskTypes;
import com.arvin.entity.Tasks;
import com.arvin.entity.Units;
import com.arvin.entity.Customers;
import com.arvin.entity.VariableTypes;
import com.arvin.entity.view.MenusView;
import com.arvin.service.manager.ServiceManager;

@Controller
public class BaseController {
	public static final Logger logger =
LoggerFactory.getLogger(BaseController.class);

	public ServiceManager serviceManager;

	@Autowired(required = true)
	@Qualifier(value = "serviceManager")
	public void setServiceManager(ServiceManager serviceManager) {
		this.serviceManager = serviceManager;
	}
```

```
    public Map<String, String> getVariableTypeList(){
        List<VariableTypes> variableTypeList =
serviceManager.getVariableTypesService().getVariableTypeList();
        Map<String, String > variableTypes = new HashMap<String, String>();
        for(VariableTypes v : variableTypeList){
                variableTypes.put(v.getVariableTypeName(),
v.getVariableTypeName());
                }
        return variableTypes;
    }

    public Map<String, String> getSystemPortList(){
        List<SystemPorts> res =
serviceManager.getSystemPortsService().getTop10SystemPortList();
        Map<String, String > systemPortList = new HashMap<String, String>();
        for(SystemPorts s : res){
                systemPortList.put(s.getPort(), s.getPort());
                }
        return systemPortList;
    }

    public Map<Integer, String> getDatabaseList(){
        List<Databases> databasesList =
serviceManager.getDatabasesService().getDatabasesList();
        Map<Integer, String > databases = new HashMap<Integer, String>();
        for(Databases d : databasesList){
                databases.put(d.getId(), d.getDatabaseName());
                }
        return databases;
    }

    public Map<Integer, String> getUnitList(){
        List<Units> unitList = serviceManager.getUnitsService().getUnitList();
        Map<Integer, String > units = new HashMap<Integer, String>();
        for(Units u : unitList){
                units.put(u.getId(), u.getUnitName());
                }
        return units;
    }

    public Map<String, String> getPageTypeList(){
        List<PageTypes> pageTypeList =
serviceManager.getPageTypesService().getPageTypeList();
```

```
        Map<String, String > pageTypes = new HashMap<String, String>();
        for(PageTypes pt : pageTypeList){
                pageTypes.put(pt.getPageTypeName(), pt.getPageTypeName());
                }
        return pageTypes;
    }

    public Map<Integer, String> getIntPageTypeList(){
        List<PageTypes> pageTypeList =
serviceManager.getPageTypesService().getPageTypeList();
        Map<Integer, String > pageTypes = new HashMap<Integer, String>();
        for(PageTypes pt : pageTypeList){
                pageTypes.put(pt.getId(), pt.getPageTypeName());
                }
        return pageTypes;
    }

    public Map<Integer, String> getTaskTypeList(){
        List<TaskTypes> taskTypeList =
serviceManager.getTaskTypesService().getTaskTypeList();
        Map<Integer, String > taskTypes = new HashMap<Integer, String>();
        for(TaskTypes t : taskTypeList){
                taskTypes.put(t.getId(), t.getTaskTypeName());
                }
        return taskTypes;
    }

    public Map<String, String> getDepartmentCustomers(int departmentId,int
positionLevel){
                List<Customers> list =
serviceManager.getCustomersService().getCustomersByDepartmentId(departme
ntId,positionLevel);
        Map<String, String > departments = new HashMap<String, String>();
        for(Customers u : list){
                departments.put(u.getUid(), u.getLastName()+",
"+u.getFirstName());
                }
        return departments;
    }

    public Map<String, String> getCustomers(){
                List<Customers> list =
serviceManager.getCustomersService().getCustomerList();
        Map<String, String > customers = new HashMap<String, String>();
```

```
        for(Customers u : list){
                customers.put(u.getUid(), u.getLastName()+", "+u.getFirstName());
                }
        return customers;
    }

    public Map<String, String> getDepartmentCustomerList(int rid){
                List<Customers> list =
serviceManager.getCustomersService().getCustomersByRoleId(rid);
        Map<String, String > departments = new HashMap<String, String>();
        for(Customers u : list){
                departments.put(u.getUid(), u.getFirstName()+"
"+u.getLastName());
                }
        return departments;
    }

    public Map<Integer, String> getDepartmentList(){
                List<Departments> departmentList =
serviceManager.getDepartmentsService().getDepartmentList();
        Map<Integer, String > departments = new HashMap<Integer, String>();
        for(Departments d : departmentList){
                departments.put(d.getId(), d.getDepartmentName());
                }
        return departments;
    }

    public Map<Integer, String> getRoleList(){
        List<Roles> roleList = serviceManager.getRolesService().getRoleList();
        Map<Integer, String > roles = new HashMap<Integer, String>();
        for(Roles r : roleList){
                roles.put(r.getId(), r.getRoleName());
                }
        return roles;
    }

    public Map<Integer, String> getCountryList(){
        List<Country> countryList =
serviceManager.getCountryService().getCountryList();
        Map<Integer, String > country = new HashMap<Integer, String>();
        for(Country c : countryList){
                country.put(c.getId(), c.getCountryName());
                }
        return country;
```

```
    }

    public Map<Integer, String> getStateList(){
        List<State> stateList = serviceManager.getStateService().getStateList();
        Map<Integer, String > state = new HashMap<Integer, String>();
        for(State s : stateList){
                state.put(s.getId(), s.getStateName());
                }
        return state;
    }

    public Map<Integer, String> getMenuRoot(){
        List<Menus> menuList =
serviceManager.getMenusService().getMenusRoot();
        Map<Integer, String > menuRoot = new HashMap<Integer, String>();
        for(Menus m : menuList){
                menuRoot.put(m.getId(), m.getMenuName());
                }
        return menuRoot;
    }

    public Map<Integer, String> getPositionLevel(){
        Map<Integer, String > positionLevel = new HashMap<Integer, String>();
        for(int i=1; i<6; i++){
                positionLevel.put(i, "Level "+i);
                }
        return positionLevel;
    }

    public Map<Integer, String> getProcessLevel(){
        Map<Integer, String > positionLevel = new HashMap<Integer, String>();
        for(int i=10; i<=100; i+=10){
                positionLevel.put(i, i+" %");
                }
        Map<Integer, String > sortMap = new TreeMap<Integer,
String >(positionLevel);
        return sortMap;
    }

        public List<MenusView> getCustomerMenusRoot(int rid){
                List<MenusView> list =
serviceManager.getMenusService().getMenusRootByRid(rid);
                return list;
        }
```

```
	public List<MenusView> getCustomerMenusLeaf(int rid){
		List<MenusView> list =
serviceManager.getMenusService().getMenusLeafByRid(rid);
		return list;
	}

	public List<Tasks> getTopFiveTaskListByReceiverId(String uid){
		List<Tasks> list =
serviceManager.getTasksService().getTopFiveTaskListByReceiverId(uid);
		return list;
	}

	public SystemVariables getSystemVariables(){
	SystemVariables systemVariables =
serviceManager.getSystemVariablesService().findSystemVariables();
		return systemVariables;
	}
}
```

```
package com.arvin.util;

import java.lang.reflect.Field;
import java.sql.Blob;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Map;

public class ObjectListToModelList<T> {
	private final Class<T> clazz;

	public ObjectListToModelList(Class<T> clazz) {
		this.clazz = clazz;
	}

	public List<String> getPropertyName(Object obj) {
		List<String> list = new ArrayList<String>();
		Field[] fs = obj.getClass().getDeclaredFields();
		for (Field f : fs) {
			list.add(f.getName());
		}
		return list;
	}

	public List<T> getModelList(@SuppressWarnings("rawtypes") List<Map>
res) throws SecurityException,NoSuchFieldException,
InstantiationException,IllegalAccessException {
		List<T> list = new ArrayList<T>();
		for (Map<?, ?> m : res) {
			T tempClass = (T) clazz.newInstance();
			List<?> propertyName = this.getPropertyName(tempClass);
			for (int i = 0; i < propertyName.size(); i++) {
				Field f =
tempClass.getClass().getDeclaredField(propertyName.get(i).toString());
				f.setAccessible(true);
				try {
					if
(f.getType().getName().equals("java.util.Date")) {
						SimpleDateFormat formatter = new
SimpleDateFormat("yyyy-MM-dd");
```

```
                                        Date d =
formatter.parse(m.get(propertyName.get(i).toString().trim()).toString());
                                        f.set(tempClass, d);
                                    }else if
(f.getType().getName().equals("java.lang.Integer") &&
m.get(propertyName.get(i).toString().trim())==null) {
                                        f.set(tempClass, 0);
                                    }else if (f.getType().getName().equals("byte"))
{
                                        Blob blob = (Blob)
m.get(propertyName.get(i));
                                        long blobLength = blob.length();
                                        int pos = 1;
                                        int len = 10;
                                        byte[] bytes = blob.getBytes(pos, len);
                                        f.set(tempClass, bytes);
                                    }else if (f.getType().getName().equals("int") &&
m.get(propertyName.get(i).toString().trim())==null) {
                                        f.set(tempClass, 0);
                                    }else if
(f.getType().getName().equals("java.lang.String") &&
m.get(propertyName.get(i).toString().trim())==null) {
                                        f.set(tempClass, "");
                                    }else {
                                        if(f.getName().equals("id") &&
f.getType().getName().equals("int")){

        f.set(tempClass,Integer.parseInt(m.get(propertyName.get(i).toString().trim(
)).toString()));
                                        }else{

        f.set(tempClass,m.get(propertyName.get(i).toString().trim()));
                                        }
                                    }
                                } catch (Exception e) {
                                        System.out.println("The Value
is:"+m.get(propertyName.get(i).toString().trim()));

        System.out.println(propertyName.get(i).toString().trim()+"["+f.getType().ge
tName()+"]" + "=>setValue() error!");
                                    }
                            }
                            list.add((T) tempClass);
                    }
```

```
        return list;
    }
}
```

Model

```
package com.arvin.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="country")
public class Country {
	@Id
	@Column(name="id")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;
	private String countryName;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getCountryName() {
		return countryName;
```

```
	}
	public void setCountryName(String countryName) {
		this.countryName = countryName;
	}
}
```

```
package com.arvin.dao.impl;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.transform.Transformers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import com.arvin.dao.CountryDao;
import com.arvin.entity.Country;
import com.arvin.entity.Page;
import com.arvin.util.ObjectListToModelList;

@Repository
public class CountryDaoImpl implements CountryDao{

	private static final Logger logger =
LoggerFactory.getLogger(CountryDaoImpl.class);

	@Autowired
	private SessionFactory sessionFactory;

   public void setSessionFactory(SessionFactory sessionFactory) {
      this.sessionFactory = sessionFactory;
   }

   @Transactional
	public void addCountry(Country country) {
		// TODO Auto-generated method stub
		Session session = this.sessionFactory.getCurrentSession();
      session.persist(country);
      logger.info("Country saved successfully, Country Details="+country);
	}
```

```
    @Transactional
        public void updateCountry(Country country) {
                // TODO Auto-generated method stub
                Session session = this.sessionFactory.getCurrentSession();
      session.update(country);
      logger.info("Country updated successfully, Country Details="+country);
        }

    @Transactional
        public void deleteCountryById(int id) {
                // TODO Auto-generated method stub
                Session session = this.sessionFactory.getCurrentSession();
                Country c = (Country) session.load(Country.class, new Integer(id));
      if(null != c){
         session.delete(c);
      }
        }

    @Transactional
        public List<Country> getCountryListByPage(Page page) {
                // TODO Auto-generated method stub
        String sql="",tsql="";
        sql="select * from Country limit
"+page.getPageIndex()+","+page.getPageSize();
                tsql="select count(id) from Country";
        Session session = this.sessionFactory.getCurrentSession();
      Query query=session.createSQLQuery(tsql);
      page.setTotalCount(Integer.parseInt(query.uniqueResult().toString()));
      query = session.createSQLQuery(sql);
      query.setResultTransformer(Transformers.ALIAS_TO_ENTITY_MAP);
      List<Map> res = query.list();
      ObjectListToModelList<Country> oltml=new
ObjectListToModelList<Country>(Country.class);
      List<Country> back=new ArrayList<Country>();
                try {
                        back=oltml.getModelList(res);
                } catch (Exception e) {
                        e.printStackTrace();
                }
      return back;
        }

    @Transactional
        public Country getCountryById(int id) {
```

```
                // TODO Auto-generated method stub
                Session session = this.sessionFactory.getCurrentSession();
                Country c = (Country) session.get(Country.class, new Integer(id));
        return c;
          }

    @Transactional
          public List<Country> getCountryList() {
                // TODO Auto-generated method stub
                Session session = this.sessionFactory.getCurrentSession();
        List<Country> countryList = session.createQuery("from Country").list();
        for(Country c : countryList){
           logger.info("Country List::"+c);
        }
        return countryList;
          }

    @Transactional
          public boolean findCountryByName(String countryName) {
                // TODO Auto-generated method stub
                List<Country> countryList = new ArrayList<Country>();
        Query query = this.sessionFactory.getCurrentSession().createQuery("from
Country c where c.countryName = :countryName");
        query.setParameter("countryName", countryName);
        countryList = query.list();
        if (countryList.size() > 0){
          return false;
        }
        return true;
          }
}
```

```java
package com.arvin.service.impl;

import java.util.List;

import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.arvin.dao.CountryDao;
import com.arvin.entity.Country;
import com.arvin.entity.Page;
import com.arvin.service.CountryService;

@Service
public class CountryServiceImpl implements CountryService {

	private CountryDao countryDao;

	public void setCountryDao(CountryDao countryDao) {
		this.countryDao = countryDao;
	}

	@Transactional
	public void addCountry(Country country) {
		// TODO Auto-generated method stub
		countryDao.addCountry(country);
	}

	@Transactional
	public void updateCountry(Country country) {
		// TODO Auto-generated method stub
		countryDao.updateCountry(country);
	}

	@Transactional
	public void deleteCountryById(int id) {
		// TODO Auto-generated method stub
		countryDao.deleteCountryById(id);
	}

	@Transactional
	public List<Country> getCountryListByPage(Page page) {
		// TODO Auto-generated method stub
```

```
		return countryDao.getCountryListByPage(page);
	}

	@Transactional
	public Country getCountryById(int id) {
		// TODO Auto-generated method stub
		return countryDao.getCountryById(id);
	}

	@Transactional
	public List<Country> getCountryList() {
		// TODO Auto-generated method stub
		return countryDao.getCountryList();
	}

	@Override
	public boolean findCountryByName(String countryName) {
		// TODO Auto-generated method stub
		return countryDao.findCountryByName(countryName);
	}
}
```

REFERENCES


[1] A. Leonard, *JBoss Tools 3 Developers Guide,* Packet Publishing Ltd, April 2009.

[2] B. Laurie and P. Laurie, *Apache the definitive Guide 3rd*, O'REILLY, Dec. 2002.

[3] B. Rücker, *Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company*, 2 edition, CreateSpace Independent Publishing Platform, Dec. 5, 2014.

[4] B. Silver, *Bpmn Method and Style, 2nd Edition, with Bpmn Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using Bpmn 2*, Cody-Cassidy Press, Oct. 17, 2011.

[5] G. Zambon and M. Sekler, *Beginning JSP™, JSF™, and Tomcat Web Development: From Novice to Professional*, New York: Apress, 2007.

[6] *HTML & CSS: design and build websites*, Indianapolis: John Wiley & Sons, Inc., 2014.

[7] L. Sommerville; *Software Engineering, 9th*,pearson, 2011.

[8] N. S. Williams, *Professional Java for Web Applications,* John Wiley & Sons, Incorporated, February 2014.

[9] J. Duckeet, *JavaScript & JQuery internactive front-end web development*, Indianapolis: John Wiley & Sons, Inc, 2014.

[10] S. Shah, *Maven for Eclipse*, Packt Publishing, August, 2014.

[11] (2016, December 15). Spring Framework [Online]. Available: https://en.wikipedia.org/wiki/Spring_Framework.

[12] (2016, December 15). SOAP [Online]. Available: https://en.wikipedia.org/wiki/SOAP.

[13] (2016, December 15). REST [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer.

[14] (2016, December 15). Hibernate (framework) [Online]. Available: https://en.wikipedia.org/wiki/Hibernate_(framework).

[15] (2016, December 15). jQuery [Online]. Available: https://en.wikipedia.org/wiki/JQuery.

[16] (2016, December 15). BPMN [Online]. Available: https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation.

[17] (2016, December 15). Maven [Online]. Available: https://en.wikipedia.org/wiki/Maven.

[18] (2016, December 15). Model-view-controller [Online]. Available: https://en.wikipedia.org/wiki/Model-view-controller.

[19] (2016, December 15). Eclipse [Online]. Available: https://en.wikipedia.org/wiki/Eclipse.

[20] (2016, December 15). Apache Tomcat [Online]. Available: https://en.wikipedia.org/wiki/Apache_Tomcat.

[21] (2016, December 15). MySQL [Online]. Available: https://en.wikipedia.org/wiki/MySQL.

[22] (2016, December 15). Unified Modeling Language [Online]. Available: https://en.wikipedia.org/wiki/Unified_Modeling_Language.