# 绪论

## 课题研究背景

随着互联网技术的飞速发展，软件项目种类繁多，软件质量的保障显的尤为重要。然而，就目前而言，国的软件测试仍然处于起步阶段，例如一些小型的软件开发公司，软件测试人员专业程度不够，甚至没有专业的软件测试人员，更没有一套适合自己的 Bug 管理系统，有的软件公司还在使用表格来进行bug的描述，人工进行Bug的统计和追踪、统计，毫无时效性，严重影响软件开发的速度，并且难以确保软件的质量。

因此，为了对软件开发周期中发生的 bug 有一个及时有效的管理方式，开发一个Java平台的简单、灵活、实用的bug管理系统就显的非常必要和重要。

## 课题研究的现状

为了提高软件开发的进度和质量，人们对软件测试行业的重视度越来越高，除了软件测试人员越发受到各个公司的亲赖外，也涌现了很多 bug 管理系统。比如说免费开源的Mantis 这一bug 管理系统，虽然它可以满足一般bug 的追踪、处理、统计等功能，但它也存在一些问题。首先它是基于 PHP/MySQL 平台，为了使用它，作为以Java 平台为核心的软件公司就要多准备一个运行环境，其次并且界面布局过于老套，不美观，在各个尺寸的显示设备上效果不是很好，最后 bug流转比较固定，缺少一定的灵活性，参与人员在处理bug上有了分歧，也没有一个很好的沟通方式。

如何在多变、复杂的软件开发中进行Bug 管理，并针对团队的现状对bug问题单的流转进行相应的调整，让 Bug 管理系统可以更好的融入到团队开发中，从而更好的加快软件开发的速度，保证开发软件的质量，是每个软件公司都在考虑的问题。如何让 Bug 的流转更加灵活高效呢？这就需要工作流技术的支持。近几年，许多信息化应用系统都集成了工作流（Work Flow）引擎技术，它通过解析一个标准的 BPMN （Business Process Modeling Notation）文件，便能够将现实生活中复杂的流程流转集成到计算机作业中来，这样一来将大大的简化复杂的业务流转。就目前而言，工作流引擎Activiti十分流行，并且它集成的流程在线设计器也可以对业务的流转进行相应的调整。

因此开发一个Java平台的简单、灵活、实用的bug管理系统变显得十分重要，本Bug 管理系统的设计与实现，是基于Java 框架 SSH并集成了工作流引擎 Activiti 框架，提供在线的 Bug 问题单的流转设计，根据自己团队的需要，设计符合自己团队开发的的问题单流转流程。

## 课题研究的意义

决定做这个 bug 管理系统初衷，与我的一小段的实习经历有关，我曾在一家小公司从事 Java实习开发职位，因为没有专业的测试人员，也没有一个有效的 bug 管理工具，在团队软件开发的过程中，成员之间发现 bug 通过 QQ 的方式在群里交谈，对于发现的bug 没有相应的系统来对它们进行管理和追踪，从而严重的拖长了软件的交付周期，也影响了软件开发的质量。所以想要动手开发一套简单、灵活、实用基于Java平台的 bug 管理系统。

现在我们都知道了，软件开发的过程中，软件测试对软件开发的重要意义。当下软件日新月异，开发周期越来越短，用户对软件质量的要求也越来越高。传统的手工处理bug 和复杂的bug流转让参与用户要花费大量的精力去关心bug 的流转问题上，这使得工作的重心产生了偏移，而且在实际的项目开发过程中，项目组的成员并不是一成不变的，软件bug的流转可能根据项目组人员的变动，职责的变动或者其他一些原因，产生变化，这个时候为了实时追踪 Bug ，就需要调整原有的Bug 流转流程，确保流转流程的完整性和有效性。

Bug 管理系统可以高效的流转测试人员提出的 bug，在 Activiti 的调动下，将bug 流转到下一个参与用户的活动任务（Task） 节点上，每个人并不需要关心整个流程的下一个节点，只需要认真自己处理分配给自己的任务，这样缩短了软件交付周期的同时也保证了软件开发的质量。

# 主要技术

## 开发环境

* 服务器中间件：在 Java EE 7 规范下（Servlet 3.1、JSP 2.3）下开发，应用服务器中间件 Tomcat。
* 数据库：MySQL 5.7。
* 开发环境：Java EE 、Maven、Git

## 主要技术简介

### SSH

SSH，即Spring MVC 、Spring 、Hibernate，是目前Java 开发web 项目非常流行一种开源框架的组合模式。

* Spring MVC

Spring MVC框架是有一个MVC框架，通过实现Model-View-Controller模式来很好地将数据、业务与展现进行分离。Spring MVC 的设计是围绕 DispatcherServlet 展开，它是一个普通的轻量级的 Servlet。和Struts2 基于类的字段的方式获取属性值，Spring MVC 通过方法参数获取属性值更加的轻量级，也能避免不必要的线程安全问题。和 Spring 的集成也要比 Struts2 更高效和方便，Struts2 和 Spring 集成时，Spring 管理bean 的方式默认是单例，这样及其容易造成线程完全问题，通过设置将 Struts2 的 Action 修正为原型，这样会造成性能的缺失。而且由于设计问题，Struts2 利用过滤器(Filter)的方式虽然可以说很好的处理请求和响应，但对于不必要的请求与响应依然进行繁琐的递归处理，另外Struts2还存在者一些安全漏洞。相比来说，Spring MVC 作为 MVC 框架更加的轻量级、安全、高效。

图 2-1 Spring MVC 请求处理流程图

* Spring

Spring框架是一个基于 Java 开发的平台，它为你的 Java 引用提供了非常全面的技术架构的支持，让您可以专注于您的应用程序。

Spring 允许创建 POJO (plain old java object) 和 非侵入性的企业应用服务，是轻量级的容器。它的两大核心是DI（Dependency Injection ） 和 AOP(Aspect Oriented Programming)。DI，即依赖注入，通过Spring 容器来管理Java 应用程序中实例的生命周期，这样的好处就是可以再运行时，通过Spring 容器动态的创建实例，动态的为注入的对象添加新的行为。AOP，面向切面编程，可以进行功能性的扩展，例如日志记录，事物控制，权限控制等。Spring 功能十分强大。

对于Java 开发者来说，Spring 提供的便利性和优势不言而喻。首先，DI更加容易组合对象之间的关系，通过面向接口编程，可以降低系统组件之间的耦合度；AOP 可以更加容易的进行功能扩展，遵循 OCP(Open Closed Principle)。并且提供对持久层（Persistence），事物（Transaction）的支持。除此之外，对于现存的各种框架，Spring 也提供了与它们整合的方案。Spring 更像是一场战争中的统帅，贯穿着表现层、业务层、持久层，管理着所有 bean 的生命周期和组合方式。

图 2-2 Spring 框架模块结构

* Hibernate

Hibernate 是一个基于 Java开发的一种ORM （Object-Relational Mapping）数据持久化开源框架。它对 JDBC 做了轻量级的封装，通过配置可以做到操作数据库而与具体数据库无关。和其它 ORM框架的原理一样， Hibernate 框架将Java 对象间的关系转换为关系数据库表间的关联，将Java 对象的字段映射到数据库表字段上。开发人员通过调用 Hibernate API ，Hibernate把 SQL 语句传给数据库，并且把数据库返回的结果分装成对象。内部封装了 JDBC 访问数据库的操作，向上层应用提供了面向对象的数据库访问 API。如下图 2-3所示，在Java 开发过程中，Hibernate 是传统 Java 对象到数据库表之间的桥梁，通过Hibernate API对作用于对象，便可以对象或者数据进行持久化操作。

图 2-3 Hibernate 位置关系

### 工作流框架

一般而言，一个成熟的运行成熟的工作流流转程序会经历5个过程，定义、发布、执行、监控、优化。而工作流的业务流转是依赖于一个遵循 BPMN规范的描述文件。通俗来讲这个文件定义了工作流的流程，是整个工作流业务流转的关键文件。由于它是一个规范，所以任何流程设计器和符合BPMN规范的流程引擎都可以解析这个文件。而目前盛行的新一代工作流框架Activiti， 支持并扩展了BPMN 2.0。

由于Activiti 的流行和Spring 的强大，Activiti 可以很轻易的集成到Spring 中来。并且Activiti 团队专门设计了用来设计BPMN 2.0 规范的流程设计器—Eclipse Designer。此外还有Signavio 公司为Activiti 定制的基于Web 的Activiti Modeler 流程设计器。

Activiti 涉及23张表，并且表设计简洁清晰，它的持久化框架选择的是比 Hibernate 还要轻量级的 Mybatis。Activiti 采用分层设计，接口编程，并且提供了七大接口，基本满足了工作流相关的功能。

表格 2-1 Activiti 的 七大接口

RepositoryService

仓库业务接口，包含读取、部署、删除流程资源等功能

IdentifyService

身份业务接口，处理用户和用户组相关操作。

RuntimeService

运行业务接口，处理正在运行的流程实例、任务等功能

TaskService

任务业务接口，包含查询、签收、办理、指派任务等功能

FormService

表单业务接口，包含读取和流程、任务相关的表单数据功能

HistoryService

历史业务接口，包含查询历史数据，流程实例、任务、变量等功能

ManagementService

管理业务接口，查询数据库、引擎配置、作业等功能

### 权限框架

Apache Shiro 是一个 Java 安全框架。就目前而言， 使用Apache Shiro 的开发者逐渐多了起来。因为它使用起来十分的简单，和 Spring Security 相比它更加的轻量级。

Shiro 可以帮助我们完成，认证，授权，加密，会话管理，与Web 集成，缓存，但它不会去维护用户和维护权限，这些需要我们自己去设计和提供，然后通过相应的接口注入 Shiro 即可。

图 1-4 Shiro 框架模块结构

### 日志框架

SLF4J(Simple Logging Facade For Java)提供了操作日志的相应接口规范，而 Log4J(Log For Java ) 框架则是针对该规范接口的对应实现。使用 SLF4J相当于一层接口，使用设计模式中的外观模式，可以很好的分离实现，而不依赖于具体的日志实现矿建。通过默认的 log4.properties 文件，通过对日志输出级别的控制，和相应的设置，可以很容易的实现对程序运行过程中日志信息的过滤，并且可以对重要的日志信息进行文件存储。

### 布局框架

SiteMesh 是一个轻量级且灵活的Java web 应用框架，基于 Servlet 中的 Filter采用了设计模式中的装饰者模式(decorator pattern) 可以很容易的对页面进行布局。它的处理流程图如下图所示。

图 2-5 SiteMesh 布局框架请求处理流程

### 前端框架列表

* JS 框架：JQuery
* CSS 框架：Bootstrap inspinia
* 客户端校验：JQuery Validation
* 富文本：CKEcitor
* 文件管理：CKFinder
* 数据表格：jqGrid
* 对话框：jQuery jBox
* 下拉框：jQuery Select2
* 树形结构控件：JQuery zTree
* 图表控件：Chart
* webim 聊天插件：layim

# 需求分析

## 系统功能需求分析

通过分析，对Bug 管理系统进行如下设计。

为了体现软件的高内聚、低耦合，按功能模块进行划分，大致划分为如下模块：

* 系统模块：用户，角色，权限，部门 管理
* bug 模块：bug，项目管理
* 工作流模块：流程，模型管理
* 通信模块：站内通知，聊天管理

其中 Bug 管理模块为本系统的核心，在实际的开发过程中我们可以了解到，在软件测试的过程中，参与的角色可以划分为：

* 测试人员：负责提交 bug ；重新验证修复的bug
* 测试主管：负责审核新建的bug；确认bug 是否有效
* 项目经理： 负责处理有效bug
* 开发主管：负责处理有效 bug
* 开发人员：负责修复分配的 bug

根据软件测试过程中角色的划分，本系统在启动时，会提供上述分析的角色和系统管理员角色。也会提供一个平台默认的 bug 流转流程，即，测试人员发现提交bug，测试主管将有效 bug 流转到 项目经理，项目经理决定 bug 是否要被处理，若需要处理，则将bug 分配给开发人员。开发人员开始修复 bug，开发人员确认提交 bug 被自己修复后，测试人员将开始再次验证开发人员修复的 bug，若验证bug 不存在，则表明bug确实已被修复，则bug 被关闭。当然，通过在线的流程设计器，也可以对平台默认的bug 流转流程进行相应的调整。

在 bug 的流程处理过程中，可能会出现 bug 的分歧，需要及时沟通，需要简单的聊天功能。

综上分析，要求bug管理系统应努力实现如下功能：

* 细粒度的权限管理功能，保证系统的安全。
* 用户登录系统可以看到自己参与项目的bug 情况，以及待自己处理的任务等详细信息。
* 以图表的形式形象、生动的显示项目 bug 状态的分布图 和 bug 状态的趋势图。
* 在线流程设计器，新建模型流程或对已有的流程进行在线的修改优化。

## 关键业务流程

## bug 生命周期

从下图我们很直观的看出 bug 的生命周期。

图 3-1 bug 生命周周期

通过上图的分析，我们可看出 bug 的生命周期一定程度反应一次Bug 提交的过程中 bug 的状态。根据图3-1 的上图分析，并结合相应的实际情况，对其进行调整，可以得到如下 Bug 状态：

* 新建(New)：缺陷首次被提交，默认状态为新建；
* 有效(Genuine)：审核通过的缺陷；
* 分配(Assign)：缺陷被分配给处理缺陷的成员；
* 打开(Open)：缺陷被分配的成员签收；
* 检测(Test)：缺陷被修复，待验证；
* 验证(Verified): 缺陷被成员验证；
* 重新打开(Reopened)：验证后，缺陷依然存在；
* 拒绝(Rejected)： 缺陷被拒绝；
* 延期(Deferred)：此次不修复该缺陷

### 3.2.2. bug 类型

在本系统中，我们将bug 类型划分为

* bug
* 改进
* 任务
* 需求

### 3.2.3. bug 优先级

在本系统中，依据 bug 的严重程度将bug 划分为如下优先级

* 紧急：非常严重 bug，需要立即解决，若不解决，基本功能无法使用，测试没

有办法进行。

* 高：严重bug
* 普通：常见的 bug
* 低：建议类 bug

### 3.2.4. BPMN 流程定义图

本系统在初次启动，会部署Bug 管理系统中默认的 bug 的流转流程，具体流程定义如图2-3所示

图 3-2 平台默认的 bug 流转流程图

## 系统用例图

### 系统模块

本模块由主要由用户管理，角色管理，权限管理，部门管理，字典管理，个人文件管理功能组成。

本平台初始化系统管理员拥有本系统最高权限，即可以对用户，角色，权限，部门，字典，文件进行管理。本平台普通角色成员，可以进行个人信息的管理，例如查看个人信息、修改个人信息、修改密码，个人文件管理，而项目经理在此基础之上，可以对用户进行一定程度上的管理。

图3-3系统模块用例图

### bug 模块

本模块由项目管理 和 bug 管理组成。

项目管理包含项目创建、查看项目信息、编辑项目、给项目组分配人员、项目统计。 bug 管理包含 bug 提交、bug 统计、bug 任务分配、bug追踪、bug 查询。

项目经理负责创建项目及其版本号，测试人员选择项目及其对应的版本号提交软件测试过程中发现的 bug ，工作流模块的调度之下，bug 的任务流转到项目组中的各个成员。项目组成员选择对应的项目可以看到项目的基本信息，项目的bug 统计分布、分配给我的任务、我提交的bug、我参与的bug 流程。

图3-4 bug 模块用例图

### 工作流模块

本模块由流程管理和模型管理两大部分组成。

模型管理包含新建模型、编辑模型、部署模型、导出模型、删除模型。流程模型包含挂起、删除、转化为模型、部署流程、删除运行中流程。

管理人员和项目经理通过新建模型，或是修改已有的模型，然后使用 Activiti 的在线流程编辑器，在本系统就可以新建 bug 流转流程或是对已有的 bug 流转流程进行修改。通过导入模型功能，导入模型进行在线编辑，然后进行部署，将模型转换为流程。通过对运行中的流程进行挂起，然后将其转化为模型，对其进行模型编辑，对模型进行调整和优化后，将其部署并转化为流程，对 bug 进行流程流转。

图 3-5 工作流模块用例图

### 通信模块

本模块分为站内通知和简单的实时聊天功能。

项目经理可以通过通知功能下达一些 bug 通知和要求，以及一些文档。平台用户也可以通过实时聊天功能，在对 bug 的处理过程中进行 bug 的沟通和交流。

图 3-6通信模块用例图

# 总体设计

## 系统功能框架

图 4-1 系统功能框架图

## 数据库模型设计

本 bug 管理系统共有 张表，除去工作流相关 23 张表外，和本系统业务相关的表有14张表，分析设计实体及其实体关系如下。

用户-角色：一个用户有多个角色，一个角色有多个用户。

用户-部门： 一个部门有多个用户，一个用户隶属于一个部门。

部门-部门：一个用户有多个下级部门，一个下级部门只有一个直接上级部门。

角色-权限： 一个角色拥有多个个权限，一个权限可以分配给多个角色。

项目-用户：一个项目里有多个用户参与，一个用户参与多个项目。

bug-项目：一个缺陷属于一个项目，一个项目有多个缺陷。

用户-用户：一个用户有多个好友， 一个用户被多个人加为好友。

用户-bug ：一个用户报告多个 Bug，一个 Bug 只能被一个用户报告。

项目-版本：一个项目有多个版本，一个版本只对应一个项目。

通知-用户：一个通知可以被多个用户接收，一个用户可以发送多个通知。

图 4-2 E-R 图

## 数据库表设计

* sys\_user 表是用来保存用户的基本信息

表格 4-1 用户表

* sys\_role 表是用来保存角色的基本信息

表格 4-2 角色表

* sys\_office 表是用来保存部门的基本信息

表格 4-3 部门表

* sys\_menu 表是用来保存权限的基本信息

表 格 4-4 权限表

* sys\_user\_role 表是用来保存用户角色关系

表 格 4-5 用户角色表

* sys\_role\_menu表是用来保存角色和权限关系

表 格 4-6 角色权限表

* sys\_role\_office 表是用来保存角色和部门关系

表 格 4-7 角色部门表

* sys\_user\_friend 表是用来保存用户好友关系

表 格 4-8 用户好友表

* sys\_dict 表是用来保存字典的基本信息

表 格 4-9 数据字典表

* oa\_notify 表是用来保存通知信息

表 格 4-10 通知表

* oa\_notify\_record 使用来保存通知关系信息

表 格 4-11 通知联系表

* bug 表是用来保存bug 信息

表 格 4-12bug 内容表

* bug\_project 表是用来保存项目信息

表 格 4-13 项目表

* bug\_version 表是用来保存项目版本信息

表 格 4-14项目版本表

* ACT\_GE\_BYTEARRAY用来保存资源部署时相关二进制数据信息。

表 格 4-15 资源内容表

* ACT\_GE\_PROPERTY 资源属性表，保存Activiti 引擎相关属性的数据

表 格 4-16 资源属性表

* ACT\_HI\_ACTINST 表用来保存历史节点数据信息，和HI\_TASKINST的区别在于 taskinst 只记录 usertask 内容

表 格 4-17 历史活动实例表

* ACT\_HI\_ATTACHMENT 表用来保存历史附件信息

表 格 4-18 历史活动附件表

* ACT\_HI\_COMMENT保存历史意见信息

表 格 4-19历史意见表

* ACT\_HI\_DETAIL记录历史流程详细信息

表 格 4-20 历史详情表

* ACT\_HI\_IDENTITYLINK 记录历史流程参与者信息

表 格 4-21 历史参与人员表

* ACT\_HI\_PROCINST记录历史流程实例信息

表 格 4-22 历史流程实例表

* ACT\_HI\_TASKINST 记录历史任务实例信息

表 格 4-23 历史任务实例表

* ACT\_HI\_VARINST表用来保存历史变量信息

表 格 2-24 历史变量表

* ACT\_ID\_GROUP 记录用户组信息记录

表 格 4-25 用户组信息表

* ACT\_ID\_INFO 记录用户扩展信息记录

表 格 4-26 用户信息扩展表

* ACT\_ID\_MEMBERSHIP 记录用户与组关联信息

表 格 4-27 用户与用户组关系表

* ACT\_ID\_USER 表保存用户信息表

表 格 4-28 用户信息表

* ACT\_RE\_DEPLOYMENT 记录资源部署信息

表 格 4-29 部署信息表

* ACT\_RE\_MODEL保存流程设计器设计流程定义后的流程设计模型记录。

表 格 4-30 流程设计模型部署表

* ACT\_RE\_PROCDEF 记录资源流程定义记录

表 格 4-31 流程定义数据表

* ACT\_RU\_EVENT\_SUBSCR记录运行时事件订阅

表 格 4-32 运行事件订阅表

* ACT\_RU\_EXECUTION记录运行流程执行实例

表 格 4-33运行流程执行实例表

* ACT\_RU\_IDENTITYLINK 记录运行时流程人员

表 格 4-34 运行流程人员表

* ACT\_RU\_JOB记录运行时定时任务

表 格 4-35 运行定时任务表

* ACT\_RU\_TASK 记录运行任务信息

表 格 4-36 运行任务表

* ACT\_RU\_VARIABLE 记录运行流程变量信息

表 格 4-37 运行流程变量表

# 详细设计

## 系统模块

### 用户管理

* 修改用户名密码

注册用户进入系统，进入「个人资料」界面，在注册信息面板上，点击「编辑」，选择「更换密码」，显示该界面，然后根据界面信息提示，输入信息，确认信息无误后，点击「确定」，提示修改密码成功。

图 5-1 修改密码事件流

### 角色权限管理

* 权限设置

管理员登录系统，进入「角色列表」页面，通过点击「权限设置」，该界面显示所有权限列表，权限列表进行勾选，点击「确认」保存权限设置。

图 5-2 权限设置事件流

* 移除用户

管理员登录系统，在「角色列表」页面，通过点击「分配用户」，该界面显示已经分配的角色用户列表信息，通过点击该页面「移除」可移除指定用户。

图 5-3 移除用户事件流

* 分配用户

管理员登录系统，在「角色列表」页面，通过点击「分配用户」，在弹出的页面上显示已经分配的角色用户列表信息，通过点击「添加人员」按钮可以再添加新的用户

图 5-4分配用户 事件流

### 字典管理

本系统使用字典管理。可以方便的管理数据库中表中某些字段的类别，比如 bug 表中 bug 状态字段、bug 类别字段、bug 优先级别字段可以在字典表中找到对应的类别标识，便于统一管理，这是对于数据库层面来说，对于 Java 编程的方式来说，所有的数据字典表在设计上，可以通过统一的方法，来获取每一个类别相关数据。

* 根据类型和值获取值
  + DictUtils.getDictLabel(String value, String type, String defaultValue)
  + ${ fns:getDictLabel (value, type, defaultValue)}
* 根据类型和标签取值
  + DictUtils.getDictValue(String label, String type, String defaultLabel)
  + ${fns:getDictValue(label, type, defaultValue)}
* 根据类型获取值
  + DictUtils.getDictList(String type)
  + ${fns: getDictList(type)}

## Bug 模块

### bug 管理

* 新建问题

测试人员登录系统，再左侧导航栏中，选择「新建问题」，进入该界面，在新建问题界面输入 bug 详情，当测试人员确认发表时，系统提示数据库保存bug 信息。

图 5-5 新建问题事件流

* 我的任务

平台用户登录系统，选择「我的任务」，进入该界面，选择一条任务记录，点击「处理」，浏览任务，发表自己的意见，当用户确认提交时，系统提示数据库保存bug 处理信息。

图 5-5我的任务 事件流

### 项目管理

* 项目添加成员

项目经理登录系统，在「项目列表」页面，通过点击「项目组成员」，在弹出的页面上显示已经项目下成员列表信息，通过点击「添加成员」按钮可以再添加新的用户。

图 5-6 项目添加成员ß 事件流

## 工作流模块

### 模型管理

* 新建模型

管理员登录系统，进入「模型列表」页面，通过点击「新建模型」，显示该界面，根据界面提示信息，输入模型信息，点击「提交」保存模型。接着页面跳转到在线流程设计器界面，进行自定义流程设计。

图 5-7 新建模型 事件流

### 流程管理

* 文件部署流程

管理员登录系统，进入「流程部署」页面，根据界面提示信息，选择部署文件，点击，系统提示转换成功。

图 5-8文件部署流程事件流

## 通信模块

### 站内通知

* 发送通知

用户登录系统，进入「通知列表」页面，通过点击「添加」，根据界面提示信息，填写信息并选择被通知成员，点击「确认」，系统提示数据库保存通知信息。

图 5-9 发送通知 事件流

### 简单实时聊天

* 聊天

平台用户登录系统，便可看到 聊天功能界面。点击用户头像，在对话框中输入聊天信息。

图 5-10移除用户 事件流

# 系统实现

## 系统模块

### 用户管理

* 用户登录

用户登录通过 Shiro 进行管理，未登录的用户若要进行有效资源的访问，会跳转到登录的页面；登录的用户只能访问自己有权限的资源，没有权限的资源，用户无法访问，会报异常。并且密码经过 SHA-1 进行加密。

图 6-1登录页面

### 角色权限管理

* 系统设置->角色管理->权限设置

图 6-2 权限设置

* 系统设置->角色管理->分配用户，之后可以移除，也可以添加成员。

图 6-3 角色分配用户

### 字典管理

* 系统设置->字典管理->添加

图 6-4 数据字典添加功能

### 文件管理

* 系统管理->文件管理

图 6-5 文件夹管理

## Bug模块

### bug 管理

* 新建问题

图 6-6 新建问题

### 项目管理

* 参与项目详情页

图 6-7项目信息

* 项目统计

本系统所有项目统计都是通过 Chart.js 来实现。前端页面通过发送 Ajax 异步请求，根据不同的需求编写不同的SQL查询语句，并结合 Chart.js 中图表类型的数据结构，将返回的结果封装成的 json 数据格式，返回给 Chart.js 的渲染图表的初始化构造函数中 。

统计项目所有 bug 状态分布和bug 趋势，项目成员在导航栏上选择自己参与的项目，在统计部分，通过 Chart.js 图表显示 bug 状态分布。

图 6-8统计 指定项目的bug 指定状态一段时间内每天数量趋势 SQL

图6-9 bug 状态趋势图

图 6-10 bug 状态分布图

## 工作流模块

### 模型管理

* 新建模型

新建模型，点击「提交」后，页面会跳转到在线设计流程定义页面，设计完毕后，点击「保存」图标进行保存。到模型管理列表，点击「部署」，对其进行部署。

图 6-11 添加模型

图 6-12 在线工作流设计器

### 流程管理

* 部署流程

图6-13上传文件部署流程

## 通讯模块

### 站内通知

向指定人员发送通知

图 6 -14发送通知

接受人登录后可以看到，任务提醒，点击相关通知记录，可查阅通知内容，并下载相关附件。

图6-15 接收到通知

### 简单聊天

在 Bug 的解决过程中，可能会出现 一些 无法解释清楚的bug，为了能够沟通清楚，这样就可以通过这个简单的即时聊天功能进行沟通。

图6-16 简单聊天

# 总结与展望

## 系统介绍

本 Bug 管理系统基于了多个优秀的开源项目和开源框架，吸取它们的经验，对他们进行整合封装而成的简单实用的 Bug 管理系统。

本系统是以 Spring Framework 为核心容器，Spring MVC 为展示层模型视图控制器，Hibernate 及 Mybatis为数据校验和数据访问层，Apache Shiro 为权限授权层，Activiti 为工作流引擎。系统主要包括基本系统模块、bug 模块、工作流模块和简单通信模块。前端界面采用简洁、大气、美观的响应式框架 Bootstrap 中主题 INSPINIA(Responsive Admin Theme) 。采用分层设计，双重校验、密码加密、访问验证，使用 Maven 做项目管理，Git 作为代码管理工具。

本Bug 管理系统主要用于在互联网公司中确保项目开发中软件的质量，并及时的反馈和统计项目中遇到的 bug。采用Activiti这个当下十分流行的工作流引擎框架技术，并在系统中集成Activiti 在线流程设计器。实现 bug的流转流程，和在线部署和优化 bug 流转流程。

该系统相比较于其他系统的优势

* 设计上采用模块设计，分层设计（展示层、业务逻辑层、数据访问层），高内聚低耦合，结构清晰明了。
* 采用 Activiti 工作流技术，实现复杂的 bug 流转流程，并提供在线修改 bug 流转流程的功能。
* 利用 Chart.js 生成形象丰富的图表记录，直观的展现项目的bug 状态分布 和走势。
* 采用简单、大气、美观的响应式前端框架 Bootstrap ，PC、手机、平板都可进行适配。
* 利用了 layim 插件，一款 webim 聊天插件 ，实现简单的实时通讯，可以实时和成员进行沟通。

## 系统存在不足

由于自己的开发经验，开发水平有限，又有加之设计与实现该系统的时间有限，Activiti 这个强大工作流框架的功能并没有完全而又灵活的应用到系统中来。在用户提交缺陷的这个页面上，bug 图片的上传，标记需要用户自行标记，带来诸多不便。

## 改进方案

针对不足，有两种方案，可以减少用户不必要的操作。

* 方案1

可以开发一个浏览器插件集成于本系统中，登录后，可以进行截图，标注，并直接上传至个人用户文件夹中。

* 方案2

现在移动设备非常流行，几乎人手一台，可以在 bug 提交页面上附上该bug 二维码。

手机通过扫描该二维码，进入系统的接口，登录后，可以对发现的bug进行拍照直传，或者从本地相册中上传图片到该bug 页面。

经过资料调查，这两中方案技术技术上都可行。

# 致谢

经过3 个多月的毕业设计终于就要结束了，经过这段时间学习，分析，设计，实现，最终基本实现了预期的功能。在这里，要感谢陈义老师对我提供的耐心指导、及时的帮助，让我能够顺利完成毕业设计。

大学四年的生活就要结束了，在这个自由的地方，不仅学到了很多专业知识，锻炼了独立思考，解决问题的能力，也明白了团队合作的重要性，更重要的是，我在大学里面找到了自己喜欢做的事情。感谢大学的这段时光，感谢学校为我们提供的学习环境和学习资源，感谢每一位老师的耐心教导，让我成为一名合格的学生。