·

本科生毕业设计论文

**题 目： 关于增删改查的快速实现 .**

**英文题目： JavaWeb Framework for CRUD .**

**学生姓名： 沈吹风 申请学位门类：工学**

**学号： 201320181405**

**专业： 软件工程**

**学院： 软件学院**

**指导老师： xxx 职称： 副教授**

**二〇一七年五月 十 日**

摘要

本论文描述的是一个基于JavaEE编程框架的设计与实现。该框架是针对于应用的后台管理系统封装的JavaEE编码框架。该框架的最大亮点为能够通过XML配置实现大部分的增删改查，能够通过JS回调，Java反射方法对配置的增删改查进行扩充。部署本框架之后还能够实现自带的用户登陆功能，操作菜单配置功能，基于角色的菜单权限控制功能。

本编码框架的设计灵感来源于公司现有的编码框架。相比较而言，该框架继承了原公司框架的思想，并舍弃了其在编码上的实现，相当于是重新开发而成，在功能上也进行了适当的扩充。

该框架目标是尽量减少开发人员重复的工作量，从而能够将更多的精力集中在软件开发的其他过程中。经过本人在公司的长期实践，我发现只要是涉及到数据库操作的系统，都可以拆分成为增删改查的实现，所以简单快速的增删改查是提高编程效率的关键。

【关键字】：快速增删改查 JavaEE框架

Abstract

This paper describes the design and implementation of a JavaEE based programming framework. The framework is a JavaEE coding framework encapsulated in a admin port of an application. The biggest highlight of the framework is that CRUD options can be achieved through the XML configuration , and can be extended by JS callback, Java reflection method. After the deployment of the framework , It will give you the base functions of user login and role based access control.

The coding framework is inspired by the company's existing coding framework. In contrast, the framework inherited the original framework's thinking , reimplements and extends it.

The goal of the framework is to minimize the workload of developers , so that they can focus more on other processes of software development.

[Keywords]: Simple CRUD , JavaEE Web Framework

目 录

[摘要 I](#_Toc484518337)

[Abstract II](#_Toc484518338)

[第 1 章 绪论 1](#_Toc484518340)

[1.1 框架开发背景 1](#_Toc484518341)

[1.2 框架开发意义 1](#_Toc484518342)

[第 2 章 开发需求分析 2](#_Toc484518343)

[2.1 主要需求 2](#_Toc484518344)

[2.2 编写环境 2](#_Toc484518345)

[2.3 Java类库依赖 2](#_Toc484518346)

[第 3 章 开发前置研究 4](#_Toc484518347)

[3.1 Java Spring框架的研究与应用 4](#_Toc484518348)

[3.1.1 简介 4](#_Toc484518349)

[3.1.2 Spring的核心思想 IoC 4](#_Toc484518350)

[3.1.3 Spring的核心思想 AOP 5](#_Toc484518351)

[3.1.4 Spring框架的基本原理 5](#_Toc484518352)

[3.2 JavaEE SpringMVC框架的研究与应用 7](#_Toc484518353)

[3.3 Java Hibernate ORM思想的研究 8](#_Toc484518354)

[3.3.1 什么是ORM 8](#_Toc484518355)

[3.3.2 Hibernate ORM 8](#_Toc484518356)

[3.4 JavaEE中SiteMesh插件与装饰者模式的研究与应用 10](#_Toc484518357)

[3.4.1 Sitemesh简介 10](#_Toc484518358)

[3.4.2 Sitemesh原理介绍 12](#_Toc484518359)

[3.4.3 Sitemesh在本框架中的应用 13](#_Toc484518360)

[第 4 章 框架功能性设计 15](#_Toc484518361)

[4.1 框架基本设计 15](#_Toc484518362)

[4.2 用户登陆设计 15](#_Toc484518363)

[4.2.1 模型设计 16](#_Toc484518364)

[4.2.2 用户登陆的页面 17](#_Toc484518365)

[4.3 权限控制设计 17](#_Toc484518366)

[4.3.1 模型设计 17](#_Toc484518367)

[4.3.2 权限控制功能设计 18](#_Toc484518368)

[4.4 增删改查方案 18](#_Toc484518369)

[4.4.1 概述 18](#_Toc484518370)

[4.4.2 基本功能 19](#_Toc484518371)

[4.4.3 原型设计 19](#_Toc484518372)

[第 5 章 框架的详细设计以及实现 21](#_Toc484518373)

[5.1 框架基本配置 21](#_Toc484518374)

[5.2 用户登录 26](#_Toc484518385)

[5.2.1 用户登录流程设计 26](#_Toc484518386)

[5.2.2 用户登陆的控制 27](#_Toc484518387)

[5.3 增删改查 28](#_Toc484518388)

[5.3.1 模型设计 28](#_Toc484518389)

[5.3.2 模型实现 30](#_Toc484518390)

[5.3.3 XML到Java对象之间转换的实现 30](#_Toc484518391)

[5.3.4 GeneralController实现逻辑 31](#_Toc484518392)

[5.3.5 XmlQueryDefineService实现逻辑 36](#_Toc484518393)

[5.3.6 视图层和前端实现 40](#_Toc484518394)

[5.4 权限控制 40](#_Toc484518395)

[5.4.1 菜单权限 40](#_Toc484518396)

[5.4.2 按钮权限 41](#_Toc484518397)

[结束语 42](#_Toc484518398)

[致谢 43](#_Toc484518399)

[参考文献 44](#_Toc484518400)

# 绪论

## 框架开发背景

当前Java语言在编程领域中已经处于发展中的巅峰。因为Java开源的特性，其在各大软件社区都是受开发者热烈欢迎的语言，各种优秀的类库也是踊跃而出。伴随着这股潮流，国内大量兴起以Java语言进行开发的软件公司。

在软件公司的实习过程中，发现一个优秀的编码框架在软件开发过程的编码阶段的影响是非常重大的。结合当前公司使用过的框架优缺点，本人决定重新开发一套简而有效的JavaEE编码框架。

## 框架开发意义

增删改查是一个系统最基本的管理功能，可以说大部分的系统功能是由增删改查的基本元素构成的。一个封装好的增删改查框架能够大幅度的提高编码人员的工作效率。

本框架开发出来的最大目标还是简化开发人员的编码工作量。目前的大量框架在编码上的繁琐程度已经达到了一个编码人员为之厌烦的地步，伴随着应用的业务逻辑逐渐扩充，大量的重复编码，无效编码也无时不刻拖累着开发人员的脚步。而规范编程，约定接口等在繁琐的劳动中也被弃而不顾。开发出这个框架，也是希望使用这个框架的人能够遵守规范，提高效率。

# 开发需求分析

## 主要需求

本框架准备针对开发人员中使用最多的增删改查需求进行技术性封装，使开发人员能够在尽量少编写代码的情况下完成复杂的增删改查业务开发。具体需要封装的功能如下

1. 列表查询（包括分页，条件查询，字段排序）
2. 新增功能（包括外键连带）
3. 删除功能
4. 修改功能

其他的通用非硬性需求

1. 用户管理（包含后台登陆）
2. 角色管理
3. 菜单管理
4. 基于角色的菜单权限控制功能。

## 编写环境

Java 开发工具包版本：1.8

集成开发环境：Eclipse，Intellij IDEA

代码托管/版本控制：Git，GitHub

项目管理工具：Maven

运行容器：Tomcat 7.0

数据库：MySql 5.6

## Java类库依赖

Spring：IOC，AOP核心，控制项目整体以及组件的生命周期。

Spring-MVC：由Spring集成，用于控制网页工程的请求与响应。

Hibernate：ORM核心，用于数据库持久层框架。

Druid：阿里巴巴提供的数据库连接池。

Slf4j-log4j2：用于日志输出。

JAXB：用于解析XML。

Gson：谷歌提供的JSON解析工具。

FreeMarker：用于替代JSP的模板渲染语言。

Sitemesh：装饰页面以减少重复代码。

Junit：用于进行单元测试。

# 开发前置研究

## Java Spring框架的研究与应用

### 简介

Spring是一个开源的轻量级Java框架，其主要的作用就是整合Java应用中使用的组件以及这些组件的生命周期管理。Spring的轻量级特征源于它不依赖于除了Java以外的任何容器，Spring自身就是一个容器，并且自身的体积小巧。

Spring基于IoC和AOP的编程思想编写，在Spring中我们可以配置各种组件放置在我们的Java应用中，同时我们也可以以一种非侵入式的方法对组件的原型、依赖和生命周期进行配置，促进了应用组件化的开发。同时，Spring也提供了强大的测试支持，我们可以轻松的使用Junit或者TestNG对Spring中的一个或多个组件进行单元测试。

Spring的出现大大简化开发人员复用代码的难度，非侵入式的配置使得我们可以在不修改源代码的前提下扩充原来组件的功能，大大简化的程序开发的过程。

### Spring的核心思想 IoC

IoC即Inversion on Control，控制反转，也被称作依赖倒置或依赖注入。它是为了降低程序耦合问题而被提出的一种全新的设计模式，是Spring框架的核心之一。

什么是控制反转？在控制反转出现之前，程序在运行过程中，一个组件的依赖往往由一个组件内部控制查找获取。而控制反转就是将查找并装配一个组件依赖的角色进行倒置，由原来的组件自己查找自己的依赖变为由容器帮组件对依赖进行查找。控制的角色由原来的组件本身变为外部的容器。这样的好处就是不需要进行侵入式修改便能改变组件依赖（如果由组件自己来装配的话想换个被装配的组建就必须修改组件装配依赖的代码，在组件是一个类库的情况下这种差异特别明显）。

在实际应用中，我们一般使用的组件（比如sessionFactory）都是由第三方提供的类库，我们是没有办法修改其中的源代码的（或者说修改源代码耗时费力），IoC的出现确实为我们开发找出了一条新的出路。

当然，以上关于IoC的说明只是其含义的很小一部分，IoC的最大作用还是减少甚至避免了在应用的开发中使用硬编码来调用服务，从而达到程序解耦和的作用。

### Spring的核心思想 AOP

AOP全称Aspect Oriented Programming，即面向切片编程。同IoC一样都是为了降低程序各部分之间的耦合度而被提出的一种编程思想。

同POP（面向过程编程）和OOP（面向对象编程）之间的关系类似，AOP和OOP并不是完全排斥的，AOP其实是在OOP的基础上进行了进一步的扩充。在开发中，OOP更加关注在建立对象模型上，而AOP的关注点则是主要在程序业务逻辑的运行上，这两者关注点不同，层面也是截然不同的。

AOP是在业务逻辑进行时的某个方法的某个阶段（方法运行之前，方法运行之后）进行统一的切面处理，一般适用于系统逻辑而不是业务逻辑处理（比如日志统计，数据库事务）。举个不恰当的例子AOP的使用就像在某类的某些方法的调用上添加了监听事件一样。因为其有同一段代码运行于多个组件切面的特点，就像加上了一道拦截网切开了程序运行逻辑，所以被命名“面向切面”。

同IoC一样，AOP也能够达到非侵入性修改组件运行逻辑的目的，毕竟他们都是为了减少程序组件的耦合度而被提出的。

在本框架中AOP应用并不多，只是在服务层的数据库操作上添加了事务管理。AOP最大作用还是用少量的代码就能切入大量的方法调用，有一种简洁的美感，一般使用于数据库事务管理或者日志时间统计，另外也有利用AOP修改Mybaties逻辑编写分页插件的实现，也有使用AOP实现业务审核逻辑的实现。

### Spring框架的基本原理

Spring之所以能够实现IoC，AOP功能，和其编写的语言Java存在密不可分的关系。Java反射功能使得解析类对象成为可能。

当然仅仅只有Java反射是不能够实现Spring的核心功能的，Spring的核心还使用到了一套强大的动态代码生成工具CGLIB。

Spring在启动时先读取xml获取要加载的Spring托管对象（Bean），当Spring实例化一个托管的对象时，Spring会生成这个对象的一个代理类，随后查找该对象的Spring依赖并注入，最后我们在程序中拿到的Spring依赖其实是Spring生成的一个代理类，正是因为代理的存在，Spring才能够实现AOP的功能。

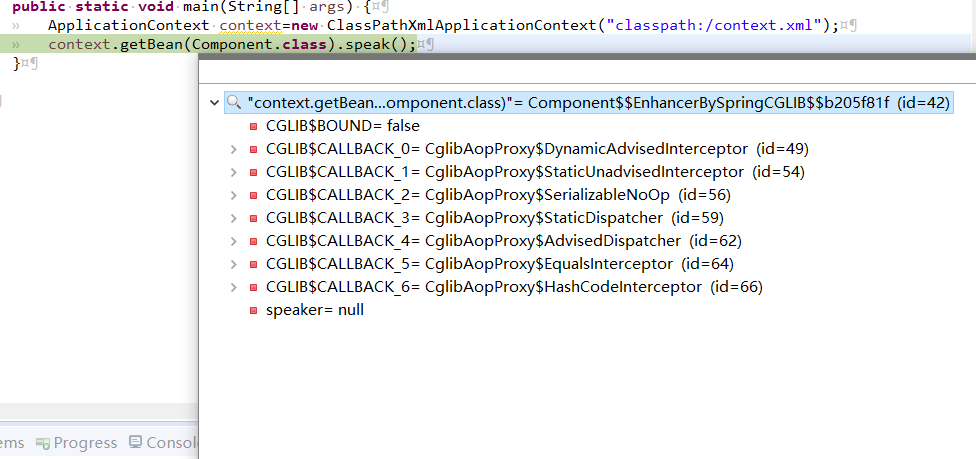


图 3‑1 Spring生成的代理类

Spring动态代理所使用的CGLIB是一套强大的动态代码生成类库。它的主要功能是生成继承或者实现于指定类的对象代理实例，因为所生成的对象实例是原类的子类，所以可以注入进依赖中。CGLIB所生成的代理类的特点是其类名中含有‘EnhancerByCGLIB’这样的关键字。本框架中所使用的Hibernate也是使用CGLIB来生成持久化托管对象的。

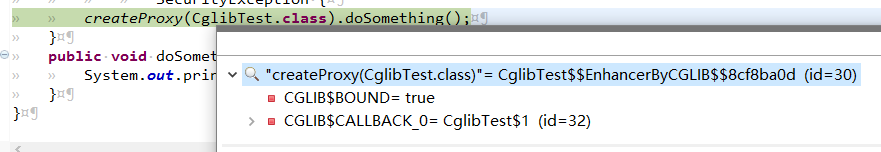


图 3‑2 代理生成的对象

其实JDK本来就有自带的动态代理接口，只不过相较于CGLIB而言，JDK动态代理仅能够代理接口的实例，而CGLIB却还能代理普通类，可见CGLIB功能非常强大。

Spring框架的基本原理当然远不止以上所说，我只是把核心的关键的进行摘要。Spring框架的能够出现最大的原因还是Java语言优秀的特性。

在本框架中Spring扮演的角色类似于一个容器，以Spring作为核心整合并控制其他组件的运行，这也是在当今的Java Web应用开发中的一个主流方法。

## JavaEE SpringMVC框架的研究与应用

Spring MVC是Spring自带的基于MVC设计模式的JavaEE控制层框架。因为它在网页应用中的特殊作用，故把它提出来作为一个单独的框架来研究。

Spring MVC由Spring自带，与Spring能够无缝集成，相比以前使用的Struts方便了不少。它继承了Spring IoC的核心思想，并将其继续发扬光大，在Controller的参数注入中体现的尤其明显。

在MVC模式中，控制层是非常核心的层次，扮演一个大脑的角色。控制层接收用户的请求，然后读取相应模型对象，将模型装入视图，发回用户，这是MVC模式中的一个基本流程。

Spring MVC的入口是一个名为DispatchServlet的Servlet，当一个请求进入这个Servlet之后，Spring MVC要做的操作就是1.查找处理器 2.调用处理器处理（这里是我们自己实现的代码）3.将处理器返回的视图和模型拼装渲染，响应客户端。当然，这里说的流程为简化流程，实际上的流程更为复杂，Spring也提供了拦截器的入口，使我们能够轻松的在处理器执行之前，处理器执行之后，视图渲染完成之后介入进行逻辑处理。



图 3‑3 Spring MVC 处理流程简图

相较于Struts，Spring MVC将请求映射到方法级别，自动将参数装箱并注入到方法的参数，是更加灵活的做法。另外SpringMVC提供的JSTL标签也极大的方便了开发人员的开发。

以下是本框架配置的spring-mvc.xml (篇幅所限略过)

## Java Hibernate ORM思想的研究

### 什么是ORM

ORM全称Object Relation Map，即对象关系映射。ORM是随着面向对象编程的发展而出现的。当前数据库主要类型是关系型数据库，而关系型数据库的表结构和面向对象的对象模型格格不入，这就需要一套能够将关系型变结构转换成面向对象数据模型的体系。

ORM的出现在数据库和对象模型之间架起了桥梁，使我们能够便捷的在面向对象程序中操作数据库，而可以不需要理解数据库的底层操作。这其实简化了开发人员开发的学习成本，使其精力集中于程序的编码上。

当然实际上ORM只是在数据库结构和数据的基础上加上了一层包装，想要使用ORM就必须有数据库中的数据，而在开发过程中数据库的优化还是需要我们访问和操作底层数据库的。

在JAVA中，当前比较流行的ORM实现框架大致有

JPA

Hibernate

Mybatis

等

其中Hibernate因为其轻量级和功能健壮性的特性成为了JAVA ORM中典型框架。

其中Mybatis因为其学习成本低，后期易优化的特点在当下大为流行。

### Hibernate ORM

Hibernet在很多开发人员眼中往往都是一个‘笨重’的框架。这个‘笨重’就体现在很多方面。

1. Hibernate的体积不小，光是引用其的依赖就要花费不小的功夫。

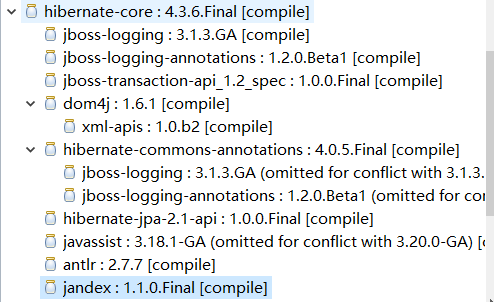


图 3‑4 本框架中Hibernate的依赖树

1. Hibernate功能十分复杂，学习使用它的时间成本非常高。
2. Hibernate对DBA十分不友好。
3. 对事务有硬性规定

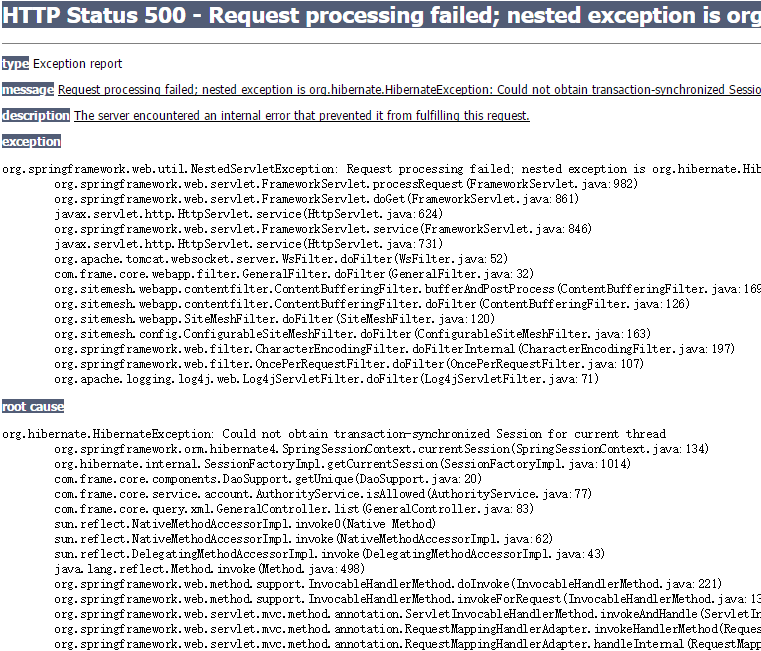


图 3‑5 不使用事务的结果

Hibernate是一套开源的，关注持久层的ORM框架。Hibernate出现于2001年，Hibernate出现很久之后，在J2EE5.0发布时，JPA才随之出现。可以说Hibernate是一个引领潮流的传奇框架。

Hibernate中的对象映射关系可以配置在对象注解中或者xml中，有了这个对象映射关系和另外配置的数据库信息之后，Hibernate就可以进行包括建表，建立表关系，数据库数据增删改查的一系列操作。Hibernate的方言设定使其可以兼容各大主流数据库，遇到特定场景的时候这一点是非常方便的。

本框架由于需要自动生成查询语句，与Hibernate特性不谋而合，故采用Hibernate作为持久层框架。

与当前流行的Mybatis相比，Hibernate的功能性要健壮的多， 当然没有那么灵活。就开发时间而言，Hibernate的前期开发时间比MyBatis少，而后期优化的开发时间比MyBatis多。就运行速度而言，Hibernate和Mybatis都差不多。不过我个人觉得Hibernate的代码比MyBatis好封装的多。

## JavaEE中SiteMesh插件与装饰者模式的研究与应用

### Sitemesh简介

Sitemesh是一个基于装饰者模式开发的JSP页面装饰器。用途是为页面添加上统一的组件，而不用每个页面都复制一套同样的代码。

最典型的一个应用就是为页面添加统一的头和菜单，通常我们都会页面面使用include标签引用同一套代码来实现这样的效果，但是这样就没有了解耦的效果。

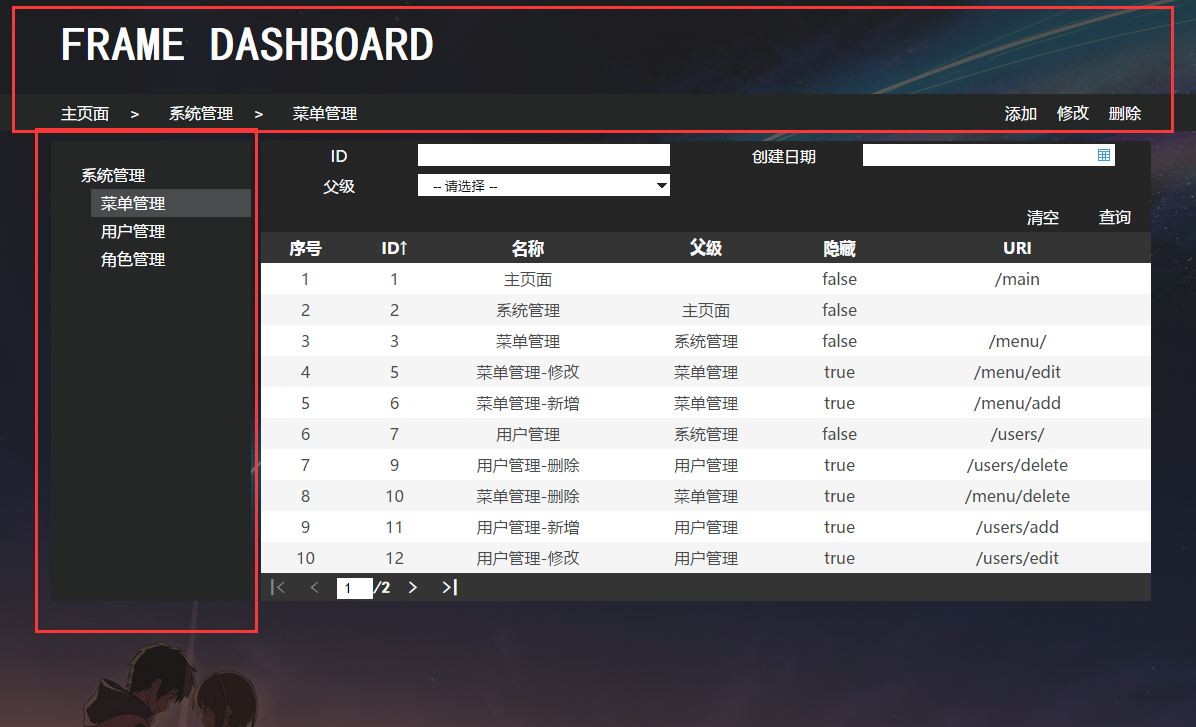


图 3‑6 本框架使用sitemesh添加的统一菜单

Sitemesh能够实现在不影响原来代码的情况下为网页动态的包装上配置中指定的组件，因为其非侵入式的特点能够很好的简化开发人员的开发，故被本框架采用。

1. 装饰器页面

这个页面逻辑比较复杂，因为要处理树级菜单的视图。另外渲染器是FreeMarker，不要被html的后缀迷惑了。

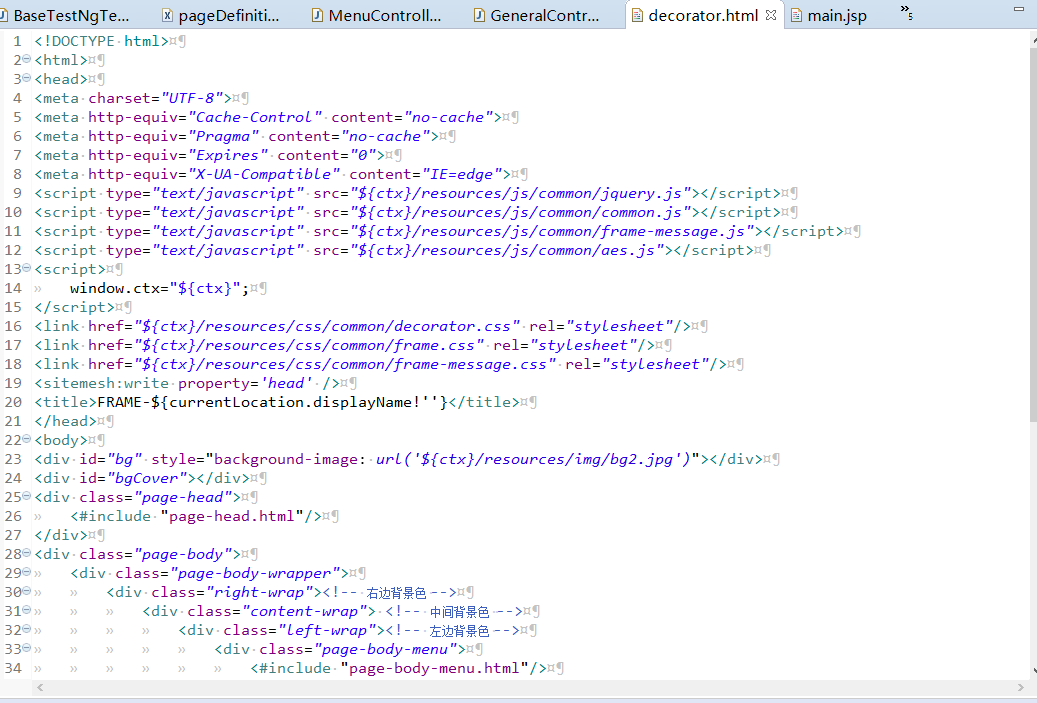


图 3‑7 装饰器页面 decorator.html

1. 被装饰的页面（JSP）

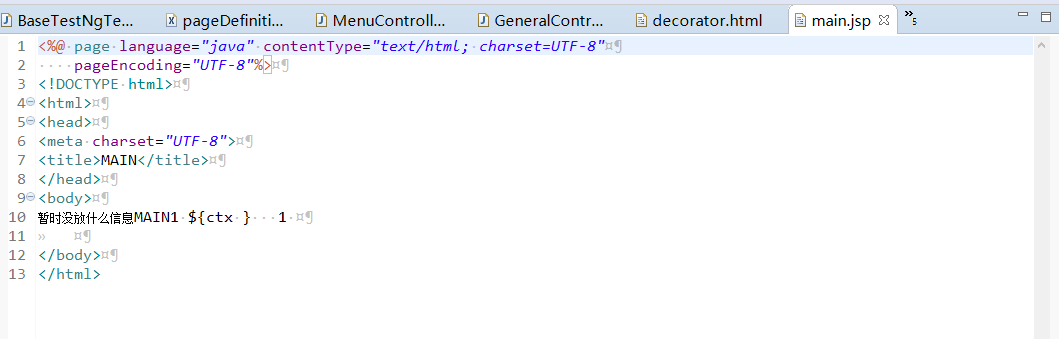
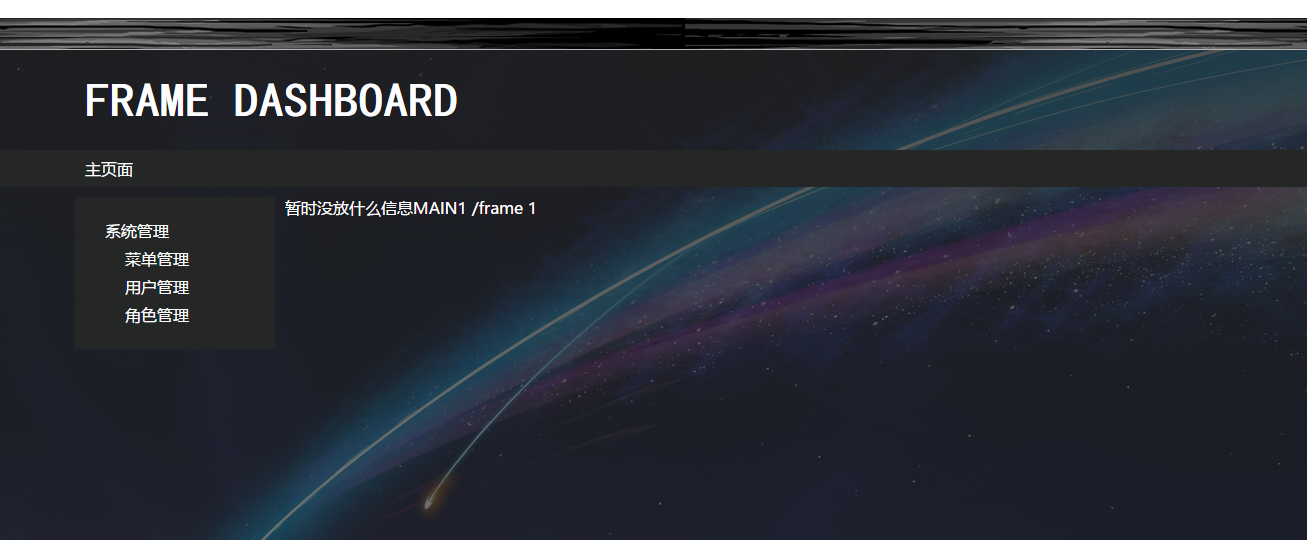


图 3‑8 被装饰的页面 main.jsp

1. 最终的效果



一般的开发人员所开发的页面为被装饰的页面，通过这个例子我们可以发现sitemesh在没有改变被装饰页面结构的情况下就完成了装饰，对开发人员而言是十分友好的，也完美的契合了装饰器模式的设计模式。

### Sitemesh原理介绍

Sitemesh配置在web.xml的过滤器中，通过截取响应（Response）并装饰达成装饰效果。大致的流程为（以上图为例）

1. 接收到请求（main.jsp）
2. 查找main页面的装饰器（找到decorator.html）
3. 把请求交给原来的处理器(main.jsp)
4. 原来的处理器处理完成，模拟出一个Request和Response发送给装饰页的处理器（decorator.html）
5. 当decorator这边处理完成之后，sitemesh就会合并这两个Response发回浏览器。



图 3‑9 sitemesh 处理图

### Sitemesh在本框架中的应用

在网上很多的博客上都有使用sitemesh的例子，只不过网上sitemesh配置的处理器都是直接为由容器托管的jsp处理器。在实践过程中我们经常遇到sitemesh和spring的兼容问题，比如想要结合由spring托管的FreeMarker视图渲染器和sitemesh，网上提供的解决方案大多是添加过滤器手动渲染装饰页[[1]](#footnote-1)。后来我在研究SpringMVC的时候找到了更好的解决方案，即可以使用Spring来托管装饰页的处理。



图 3‑10 一般的处理结构

其实通过很简答的配置就可以将装饰页交给SpringMVC。

其中，resource文件夹存放静态文件目录，不需要装饰，login页面ui另外设计不需要装饰。网页json的ajax请求因其请求MIME类型为json 会被自动过滤，而ajax请求的页面如果不需要装饰就只能使请求地址中包含ajax关键字来过滤了。

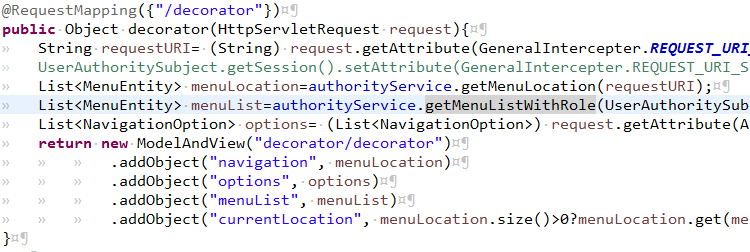


图 3‑11 Controller的处理

以上的表格中配置的装饰器的地址为“/decorator”，图片中Controller中的RequestMapping为“/decorator”，这样装饰页的视图的查找，数据处理，以及视图的渲染都为Spring完成，sitemesh只是起到一个引导并装饰的作用。



图 3‑12 本框架Sitemesh配置

# 框架功能性设计

## 框架基本设计

在工程的构建上，使用Maven，使用GitHub进行代码托管。使用着两个工具在行业中已经趋于主流。

本框架准备使用Spring+Hibernate+SpringMVC作为主干框架，其中Spring作为基本的容器对框架组件进行托管，Hibernate用为持久层的控制，SpringMVC管理HTTP请求响应的控制。

由于使用了Hibernate，数据库在本框架中变得不再重要，这里就不在赘述。这里的数据源使用了阿里巴巴出品的Druid，其特点是可以进行方便的监控。另外本框架还准备编写一套自动初始化数据库的Spring组件，但是由于一些其他方面（主要是可靠性）的原因已经弃用了。

其他方面，本框架使用FreeMarker作为页面渲染语言，另外兼容JSP和静态HTML，使用Sitemesh进行装饰，使用log4j2为实现，slf4j为接口的日志输出。

前端方面使用jQuery框架，其他控件自己封装。

这样，本框架也有了一个基本的概念。

* 本框架为一个比较重型的Java Web前后端框架。
* 本框架的适用范围为网页后台管理门户的开发。
* 本框架兼容各种主流数据库。
* 本框架希望能够尽量减少开发人员的编码量，特别是前端（当然如今网页应用的需求五花八门，想要达到这个目的路途还遥远得很）。
* 本框架使用服务端渲染视图，注重服务端的运算，在性能方面其实还要做很多优化。

另外，过度的封装必然导致自定化的不足，想要实现一些高度个性化的东西还是得需要高超的编程技巧。

## 用户登陆设计

登陆是每一个门户绕不开的坎，登陆做的不好能够毁掉整个应用。登陆虽然简单，但是要做到安全还是要注意很多事项。

1. 如何防止SQL注入。

这是一种典型的黑客技巧，防范起来非常简单，但是如果没有做好后果严重。解决措施就是使用SQL占位符。

很多网页应用使用SQL拼接来实现数据库查询，而登陆的查询语句最典型的就是“select \* from user where username=‘username’ and password=‘password’”这样的语句。那么我只要密码填写“ ’ or 1=1 or ‘’=‘ ”这样的语句就可以不使用密码进入系统。使用SQL占位符能够把用户填写的单引号等特殊符号转换成SQL转移符（类似“\’”）。

1. 如何优雅的加密。

加密方式有多种，本框架准备采用AES的前台加密后台解密以及SHA1的后台用户名密码加密的方式进行加密。即用户填写登录名密码之后将用户名密码在前端进行AES加密，数据传到后台再AES解密然后再使用SHA1对账号密码进行加密，这样密码在用户提交登陆之后从客户端到服务端再到数据库都是密文状态的，有一定的安全性。当然也有在后台不进行解密然后直接继续加密的，安全性更高。[[2]](#footnote-2)

AES是一种对称加密算法，使用一个秘钥以及偏移量对明文进行加密，可以解密。

SHA1是一种摘要加密算法，除了暴力方式以外没有解密方案。

### 模型设计

用户模型请参考用户管理设计中的模型设计。

### 用户登陆的页面



图 4‑1 登陆流程设计

## 权限控制设计

### 模型设计

以下是相关模型的ER图



图 4‑2 从用户到菜单的权限控制关系

### 权限控制功能设计

本框架准备实现基于角色的菜单权限控制功能。所谓基于角色的菜单权限控制即每一个用户都有其设定的角色，每个角色可以访问若干菜单。这样就能够规定每一个用户能访问的菜单，通过菜单的控制达到控制用户能够访问目录的效果。

其中具体目录下的按钮由于通过本框架来托管，只需要配置也能够达到操作权限控制的效果，即权限控制的层面能够达到具体业务下操作权限的底地步。另外的权限控制如可访问数据权限控制由于过于复杂，应用场景也不是非常的多本框架并没有考虑设计。

当然，这样基于菜单的权限控制也有不足，用户可以通过手动指定url的方式强行访问指定的界面。这个问题也不是没有解决办法，只要添加过滤器进行url控制即可，这样的实现方式也应该称作基于角色的URL权限控制功能。出于简单性考虑本框架暂时不做这方面的设计。

## 增删改查方案

### 概述

这里的模型过于复杂，所以模型部分移入详细设计。

要想实现封装增删改查，配置文件是必不可少的，还要考虑实现的同时能够使开发人员能够介入操作。基于以前公司接触的多套框架得到的灵感，本框架采用xml进行增删改查的配置。

### 基本功能

以下是增删改查方案的基本功能。

表格 4‑1

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 所属模块 | 具体功能 | 说明 |
| 1 | 查询 | 可配置的条件查询 | 对应hql语句中的where 条件 |
| 2 | 连表查询 | 对应hql语句中的 join 以及没有建立外键关系的 where语句交叉连接 |
| 3 | 分页 | 对应查询对象中的setFristResult和setMaxResult |
| 4 | 排序 | 对应hql中的order by字段 |
| 5 | 删除 | 删除 |  |
| 6 | 新增 | 可配置字段的新增 | 新增和修改使用同一页面。 |
| 7 | 修改 | 可配置字段的新增 |

另外，查询和修改新增的表单输入框应当能够支持多种不同的类型。

### 原型设计

1. 查询页面



图 4‑3 查询页面原型

1. 修改/新增页面

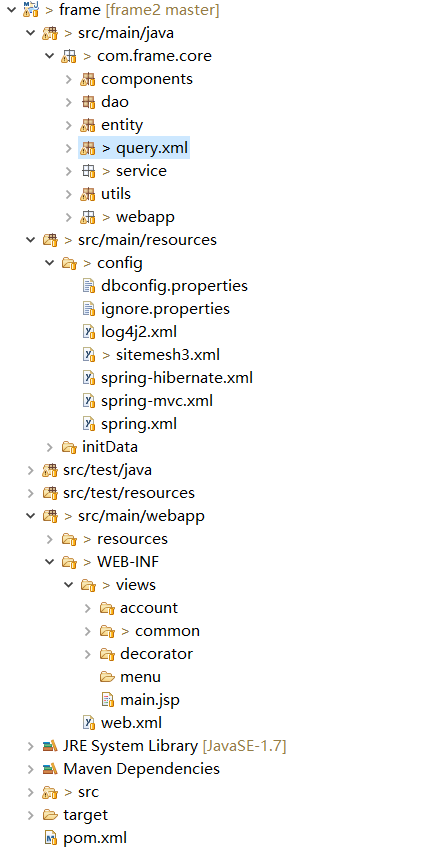


图 4‑4 新增/修改页面原型

# 框架的详细设计以及实现

## 框架基本配置

1. **整体工程文件结构概览**



这里只显示大概的工程目录，代码过于冗杂就不贴了，这里贴一下代码地址。

代码地址：<https://github.com/DeffersonCheng/frame>

1. **Gson与SpringMVC的实现**

在SpringMVC中AJAX返回的JSON对象处理一般交于Jackson处理，而在我们框架中我们希望使用Gson。我在Spring的源代码中找到了Spring对Gson的支持（org.springframework.http.converter.json.GsonHttpMessageConverter），然后利用Spring的IOC对本框架中使用的gson进行配置。如下

表格 5‑1 gsonFactory的配置

|  |
| --- |
| <bean id=*"defaultGson"* class=*"com.frame.core.components.GsonFactory"* factory-method=*"buildDefaultGson"* scope=*"singleton"*/> |

1. **统一AJAX返回对象AjaxResult设计**

AjaxResult类是一个标准AJAX返回值的结构体。一般AJAX请求返回AjaxResult就能够适应大部分情况。

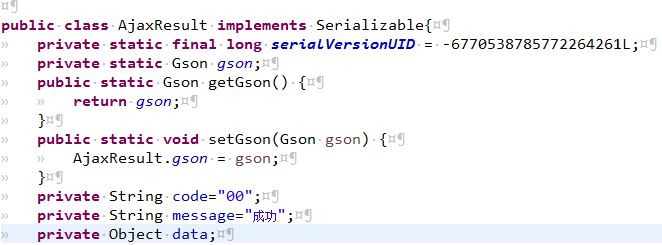


图 5‑1 AjaxResult

其中的gson对象在Spring初始化完成之后注入，为单例模式。

1. **兼容多种的视图处理器实现**

一般网页框架都配置固定一个的视图处理器，本框架因为特殊原因（FreeMarker学习成本）决定采用以FreeMarker为主，兼容jsp和静态html视图处理的适配机制。

这里有大量的实现代码略去，主要的思路为继承Spring视图处理器规范编写自定义视图处理器，通过获取返回视图的后缀名来选取特定的视图渲染器。

1. 框架组件：基于ThreadLocal的线程绑定变量ThreadBinder的实现

线程绑定变量即每一个线程独享一个实例的变量，在单一线程中可以等价于全局变量。因为JavaEE中同一个请求的处理是在同一个线程中的，所以很多情况下会将Request Response以及Session对象绑定在线程中供全局使用减少耦合。

ThreadLocal是Java中提供的的一种高级工具，Shiro使用它实现了Session的全局工具类获取，Spring使用它实现了DAO层的单例模式无状态化。

需要注意的是，由于Tomcat使用线程池处理请求，同一个线程可能会接连处理两个请求，所以如果不是每一次进入线程时覆盖线程绑定变量，会使用到上个请求遗留的线程绑定变量。另外，需要另外开线程的大量逻辑处理以及RPC都无法使用线程绑定变量。

1. **基于ThreadBinder的用户权限校验组件的实现**

这个组件名为UserAuthoritySubject，是模仿Shiro权限框架实现的，主要用于取得用户的账户信息，该组件目前来看还没有完善，其功能也只有取得Session和取出当前登录用户的UserEntity对象。

其实就是本框架中提供给开发人员用于快速获取Request，Response，Session的工具类，使用它可以达到参数传递解耦的目的。

1. **基于Hibernate的BaseEntity的实现**

在本框架中所有的实体都必须继承的数据库实体模型基类，不继承会导致没有办法使用框架增删改查的情况。

其中封装了数据库实体的一些基本属性，比如ID，创建时间等，本框架中大量的处理逻辑依赖BaseEntity。

1. **基于Spring支持的HibernateDaoSupport的实现**

这个类封装了绝大部分情况的一行查询。其中Spring部分在HibernateTamplate对象中实现，拓展部分为直接调用。

其中逻辑代码和Spring HibernateTemplate被封装在DaoSupport类中，直接能够调用的是继承DaoSupport类的GeneralDao类。该类可以作为一个通用的数据库访问类或者编写Dao类的范例来使用。

由于GeneralDao功能已经足够强大，本框架中并没有编写其他的DAO，以后说不定会有。

1. **frame.message实现**

frame.message是一个基于jQuery封装的前端消息提示组件，用途是替代js中的alert等。

这里略去实现代码。

该组件可以在被装饰页的全局中调用到，调用方法有 frame.toast(msg[,clickCallback]) 和 frame.tosatError(msg[,clickCallback]) 两个方法，以后可能会拓展一个confirm的方法。

最终实现的效果：

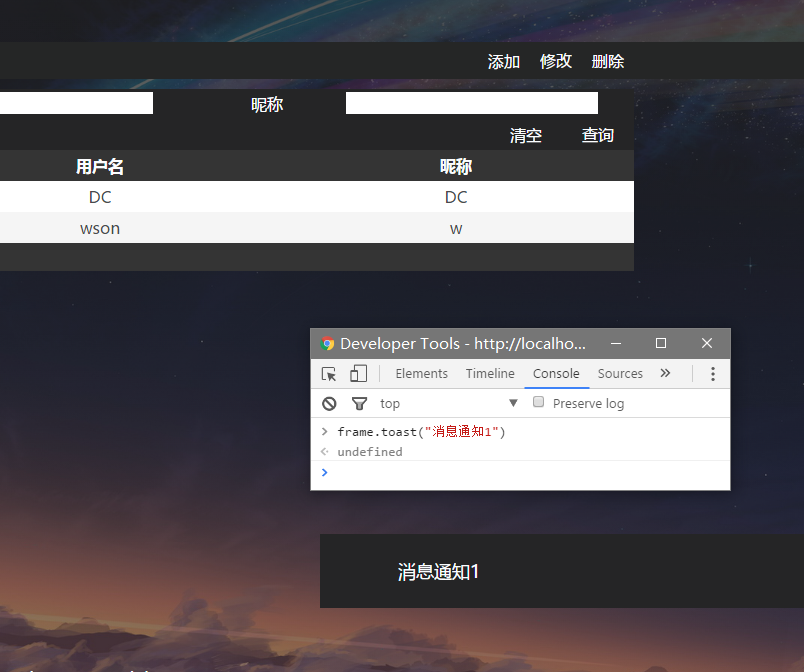


图 5‑2 最终实现的效果

由于这里大量使用css3，可能对于老旧版本的ie支持性并不良好，这里并没有进行测试。

1. **框架异常处理机制**

在Java中异常分为普通异常和运行时异常，其中普通异常在抛出时需要声明，在使用时需要强制对其处理。而运行时异常在开发人员不是主动去处理时会被默认抛出最外层。一般来说内部进行处理时抛出运行时异常可以适当简化代码，而开发对开接口或者需要被其他人员调用时可以使用普通异常。在本框架中，大部分的异常继承于运行时异常。

而在一般的应用程序中，异常一般分为业务异常和系统异常，在本框架中，一般异常都统一会显示到前台。

本框架主要处理了两种异常，一种为普通页面请求时发生的异常，一种是Ajax请求时发生的异常，我们主要是对Ajax异常做了特殊的处理。

处理逻辑为在Spring拦截器中拦截Ajax请求发生的异常，组装成特定的本框架中定义的Ajax异常协议指定格式JSON字符串返回。前端使用中使用ajaxSetUp在生成特定的Ajax异常处理逻辑。

Ajax异常协议为：

状态 Ajax 500

响应头 AJAX\_ERROR 1，Content-Type application/json;charset=utf-8

响应体 格式：JSON ，组成 code：-1，message：rootCause，data：stacktrace

注：在正式环境中，异常详细信息不应该抛至前台显示，这部分信息应当屏蔽。

另外，本框架支持通用异常的统一处理和个别业务出现异常的定制化处理。

## 用户登录

### 用户登录流程设计



图 5‑3 进入登陆页面的流程



图 5‑4 登录页的处理流程

### 用户登陆的控制

在本框架中，用户登陆控制在Spring拦截器中实现。用户登陆控制分为两种，一种为非AJAX请求登陆控制，一种为AJAX请求登陆控制，其中AJAX请求的登陆控制需要前端JS配合。

具体实现逻辑为在Spring统一拦截器中判断登录状态，然后判断是否AJAX请求，然后返回对应的信息。若是AJAX请求则返回响应头中添加SESSION过期的标志，不是则直接重定向至登陆页面。

## 增删改查

### 模型设计

以下是整个配置类的定义，部分不配置字段没有显示。

1. PageDefinition//页面定义
   1. queryDefinition：QueryDefinition//查询定义
      1. mappedClass：List<MappedClassEntry>//要查询的类
         1. mappedClass:Class<? Extends Serializable>//类名
         2. alias:String//别名
         3. join: List<JoinEntry>//连接
            1. type:String//连接类型 左连接 右链接
            2. field:String//连接的字段
            3. as:String//别名
      2. where：String//固定的查询条件 1=1 and 2=2
      3. sortBy：List<SortEntry>//排序
         1. field:String//根据字段
         2. fromAlias:String//是哪个别名的字段 可以不写
         3. order:String//升序还是降序 ASC | DESC
      4. showIndex:String//序号的表头 不填代表不要序号
      5. columns:List<ColumnDefinition> //列定义
         1. header:String//表头
         2. field:String//字段
         3. fromAlias:String//字段属于哪个别名的类，可以不填
         4. width:String//宽度 可以填百分比和px单位
         5. sortable:boolean//是否可以排序
         6. hidden:boolean//是否隐藏
         7. staticColumnData:String//如果不为空这个字段不会查库而是显示这个字符串，可以填写按钮之类的标签，注意包裹上CDATA
         8. filter:Class<?>//过滤器 如果这个字段不是基本字段需要手动设置来转换为字符串
      6. pageSize:int//一页默认个数
      7. queryConditionDefines:List<QueryConditionDefine>//查询的定义
         1. field:String//条件字段
         2. alias:String//条件的字段属于哪个别名 可不填
         3. operator:String//条件比较的操作符 大于或者小于
         4. nullable:boolean//是否能够为空 默认可为空
         5. defaultValue:String//默认查询值 可选
         6. inputType:String//输入框类型 默认TEXT 可选
         7. displayText:String//label中的字符串
         8. inputId:String//输入框的DOM id 可选
         9. staticData:String//若为下拉框可以填写JSON串定义下拉框的选项
         10. parsedData:Object//不能配置，为处理数据时暂存的字段
         11. selectTextField:String//下拉框的展示字段
         12. selectValueField:String//下拉框的值字段
         13. optionClass:Class<?>//若下拉框是数据库中查询的，可以定义这个类自动查询，以后可能会增强使其支持枚举类。
      8. requiredJsPath:List<String>//页面额外包含的js 直接写/resources/js/xxx.js之类就行
      9. requiredCssPath:List<String>//页面额外包含的css 直接写/resources/css/xxx.css之类就行
      10. navigationOptions:List<NavigationOption>//处于计划阶段，还没有实现
   2. manage：Manage
      1. field:List<ManageField>
         1. field:String
         2. defaultValue:Object
         3. inputType:String
         4. displayText:String
         5. selectTextField:String
         6. selectValueFiled:String
         7. inputId:String
         8. optionClass:Class<?>
         9. nullable:boolean
      2. beforeManage:String//填写controller中的方法名，在保存或更新之前执行，return false或者抛出异常不执行
      3. beforeManageMethod:Method//配置转为方法对象之后的字段
      4. afterManage:String//填写controller中的方法名，在保存或更新之后执行
      5. afterManageMethod:Method
      6. requiredJsPath:List<String>//页面额外包含的js 直接写/resources/js/xxx.js之类就行
      7. requiredCssPath:List<String>//页面额外包含的css 直接写/resources/css/xxx.css之类就行
   3. delete：Delete
      1. beforeDelete:String//填写controller中的方法名，在删除之前执行，return false或者抛出异常不执行
      2. beforeDeleteMethod:Method
      3. afterDelete:String//填写controller中的方法名，在删除之后执行
      4. afterDeleteMethod:Method

### 模型实现

这里略过。

### XML到Java对象之间转换的实现

我们定义了使用xml来对增删改查进行配置，所以xml模型和java模型之间的转换是必不可少的。这里我们使用jdk自带的JAXB来实现xml到模型实例的转换，其中最关键的就是@XmlRootElement注解，其他的注解用于映射xml文件中的节点以及属性与java类中的字段的对应关系。

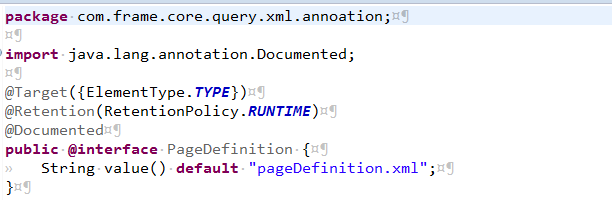


图 5‑5 PageDefinition注解

PageDifinition注解配置于Controller上，这个Controller必须继承于GeneralController，该注解用于定义Controller对应xml的路径。

这是一个XML转换成Java对象之后的承托类，用途是加载并承载并初始化PageDefinition对象。其中Unmarshaller是一个JAXB提供的反序列化工具。PageDefinitionHolder类在初始化时查找PageDefinition注解加载xml定义文件路径并初始化Unmarshaller对象。

PageDefinitionHolder.refresh方法用于刷新（加载）xml，考虑到更改xml后重启系统的时间损耗，我们这边采用基于修改时间判断的实时加载策略。

### GeneralController实现逻辑

#### 概述

GeneralController是使用增删改查框架必须继承的基类，GeneralController作为Web请求的入口处理增删改查业务逻辑。



图 5‑6 GeneralController处理逻辑及生命周期

#### 构成

GeneralController几个对象组成：

1. targetClass：Class<T>
2. LOGGER ：org.slf4j.Logger 用于打日志
3. pageHolder：PageDefinitionHolder 加载的配置
4. service：XmlQueryDefineService 处理具体的逻辑
5. authorityService：AuthorityService权限处理接口

#### 初始化

一共做了以下几个步骤

1. 判断泛型类型并初始化targetClass的值。
2. 加载PageDefinition注解配置，读取配置文件路径。
3. 使用读取的配置文件路径初始化PageDefinitionHolder对象。
4. 使用refresh函数加载配置。

#### 查询请求的流程



图 5‑7 list请求中GeneralController大致处理流程

注：这里某些流程顺序和代码不一致，因为从逻辑来书这样比较清晰。

查询功能包括基本查询，条件查询，分页，排序四个子功能，可以拼成一个请求流程来实现。虽然说整合起来逻辑比较复杂，但是经过封装之后，实现这些逻辑就可以省下来很多的时间。

#### 删除请求的流程



图 5‑8 删除请求具体流程

删除功能非常的直接简单，只不过增加的回调功能之后也显得复杂起来，因为回调功能使用了大量的反射写法。其中一些细节并没有显现在上图中，比如删除成功之后如何通知前台。

#### 增加/修改请求流程

这里将增加和修改拼合在一个页面中，做到代码尽量复用。增加/修改功能需要两个请求流程，一个是准备数据和页面（新增中的数据为初始化的数据，修改中的数据为数据库加载的顺序），一个是新增/修改完成之后保存数据的请求。



图 5‑9 准备新增/修改页面流程



图 5‑10 GeneralController的saveManage逻辑

### XmlQueryDefineService实现逻辑

这里包含了本框架实现的核心逻辑，在GeneralController中有提到，我们拆分出来描述。

#### 构成

1. LOGGER：org.slf4j.Logger 打日志
2. gson：Gson
3. defaultDataFilter：com.frame.core.query.xml.DefaultDataFilter 类型转换器，用于将各种一般类型的对象转换成字符串形式。
4. dao：GeneralDao 数据库操作类

另外还定义了一些异常，用于包裹处理中出现的异常。

DataSetTransferException 数据转换异常

QueryConditionParseException 查询条件识别异常

ManageExecuteException 执行修改操作出现的异常

#### 查询处理（list）



图 5‑11 list方法的流程

QueryHqlResolver其实就是一个工具类用于根据参数（页面传参+配置读取）生成Hql。

DefaultDataFilter负责将不同类型的数据转换成字符串。

#### 查询处理（count）

在分页查询中有一个必不可少的查询就是查询复核条件数据的总数量，用于确定总页数。



图 5‑12 totalPageCount流程

#### 删除处理

XmlQueryDefineService.delete这里利用Spring对Hibernate提供的模板进行删除。

#### 新增/修改（页面）



图 5‑13 prepareManage流程

开始的clone代码块是为了解决多线程同步问题编写，因为多线程中配置类只有一套，而页面可以同时存在多个，为了避免不同页面之间的数据干扰ManageFields使用独立的新对象。

另外准备复杂关系数据是指有一对一、一对多或者多对多这种需要下拉框控件或者多选框控件的数据准备工作。

#### 新增/修改（提交）



图 5‑14 saveManage流程

这里的代码又是非常复杂，这里做一下说明。

1. 在更新的情况下，为了防止配置文件中只配置部分字段而把原来存在的字段刷掉故采用先加载库实体然后覆盖字段的方式。
2. 含有关联关系的字段中，若关联的对象不是持久化对象更新会报错。
3. 和删除不同，这里的回调由service层完成，这个设计并不好。

### 视图层和前端实现

说明：视图层使用FreeMarker语言实现，其实我认为使用JSP自定义TAG也是一种非常不错的实现方式。因为这里的逻辑不重要所以不做很多描述了。

#### 查询页面视图

文件名：common-list.html

#### 查询页面脚本

文件名：common-list.js

#### 新增/修改页视图

文件名：common-manage.html

#### 新增/修改页面脚本

文件名：common-manage.js

## 权限控制

### 菜单权限

菜单权限主要通过控制页面左侧的菜单树的显示与否来达成权限控制的目的，实现方法是加载菜单时判断当前角色权限进行剔除。



图 5‑15 AuthorityService中对菜单权限控制的关键代码说明

### 按钮权限

本框架中一共控制了三个按钮的权限，分别是增加、修改、删除。实现原理为在GeneralController中添加按钮时对权限进行判定。



图 5‑16 在GeneralController中对按钮权限进行判断的关键代码

当然这可能不太够，之后本框架可能会拓展自定义按钮并做到自定义按钮的权限配置。

结束语

本框架原来的构想是使用JavaEE封装构建出一个伪事件响应型的网页框架，类似.NET那种，结果实现起来才发现那不太可能，很复杂，只能退而求其次，实现了这个框架。

当然这个框架还有太多太多的不足，要想真正应用到企业级的应用中还是需要更多地打磨，我也会不定期的更新代码，也希望这个框架有人能够去尝试使用，毕竟真正用起来才能发现更多的问题。

碰到的问题也很多，最典型的Hibernate关联关系配置就经常出现奇奇怪怪的问题，还有JAXB的XML转换技术使用的不够专业等，有些克服了，有些直接放弃了。代码是我在工作之余抽空写的，所谓最大的敌人还是自己，有时候自己的代码都看不懂，时间好像也不怎么充裕，这就导致了这个框架的代码十分碎片化，有些逻辑相似的地方实现却完全不同这种情况是否发生。

本次的毕业设计可以当做我这出外工作实习一年在JavaEE上的总结，我也是尽量的把学到的东西应用到本框架中。其实这个框架是有名字的，叫“frame”，虽然整篇文档都说着“本框架”。

致谢

本框架的灵感来源于本人在公司益博睿项目组使用的Hibernate框架，不过所有的实现都被放弃而采用了其他的实现方法，其实逻辑复杂度是大大增强了，不过我个人认为实现出来的效果还是不错的，特别是UI。其中的Sitemesh发现于公司的OA系统，经过研究被我掌握，在益博睿项目组也有使用。JAXB是网上搜索看到的技术，因此碎片化很严重。另外的一些概念理念是自己在长时间的编码中通过实践反向理解出来的，所以有些描述不太准确，甚至于真相不符都是有可能的。

本框架的开发基本上是由我个人独自完成的，我也愿意拿出来给大家提供参考和分享。

在这里感谢公司提供的良好环境让我看到大量的可借鉴的代码；感谢xxxx导师为我的毕业设计提供指导；另外感谢wosenyu同学在使用本框架中为我提出的建议以及提出的一些新型技术。

参考文献

1. Booch G. Object-Oriented design[J]. ACM SIGAda Ada Letters. 1982,1
2. Seung Mo Cho,Hyung Ho Kim, Sung Deok Cha etc al. A semantics of sequence diagrams [J]. Information Processing Letters 84. 2002
3. Abdellatif, T. and F. Boyer. A node allocation system for deploying JavaEE systems on Grids. 2009. Hammemet, Tunisia.
4. Kachru, S. and E.F. Gehringer. A comparison of j2ee and. net as platforms for teaching web services. 2004
5. XIONG, J., L. YAO and J. HU, Implementation of Dynamically Generating HTML WebPages by C\#. Computer and Modernization, 2007. 10.
6. 陈雄华/林开雄. Spring 3.x 企业应用开发实战[M].北京:电子工业出版社,2012-02
7. 苏翻.基于 J2EE 平台的改进 MVC 的研究及其应用[J].南京航空航天大学学报.2006.Vol.12
8. 龚睿.杨贯中.陈莉.面向 Web 服务的电子政务工作流模型研究[J].计算机工程与应用.2005.Vol.24
9. 张爱平,赖欣.在JSP中调用JavaBean实现Web数据库访问[J].计算机时代,2007
10. 仲伟和.基于JSP网页自动生成工具的设计与实现[J].科技信息(科学教研),2007
11. [捉住你的尾巴](https://jingyan.baidu.com/user/npublic?un=%E6%8D%89%E4%BD%8F%E4%BD%A0%E7%9A%84%E5%B0%BE%E5%B7%B4). B/S架构与C/S架构的区别[CP/OL]. 百度经验，2014-07-17 [2017-01]. <http://jingyan.baidu.com/article/b0b63dbfc9a2874a483070c0.html>.
12. [Three Apple](http://home.cnblogs.com/u/sinlang5778/). PHP,JAVA,NET 开发比较[CP/OL]. CSDN，2011-8 [2017-01]. <http://www.cnblogs.com/sinlang5778/archive/2011/08/10/2133190.html>.
13. [Copyvio](http://www.baidu.com/p/copyvio). JavaEE解释[CP/OL]. 百度百科，2016-08-29 [2017-01]. <http://baike.baidu.com/item/JavaEE>
14. [Java118](http://www.baidu.com/p/java118). JDK释义[CP/OL]. 百度百科，2017-03-31 [2017-01]. <http://baike.baidu.com/item/jdk/1011>
15. Java118. Eclipse释义[CP/OL]. 百度百科，2015-11-29 [2017-01]. <http://baike.baidu.com/item/eclipse/61703>
16. [07806919d](http://www.baidu.com/p/07806919d). Github释义[CP/OL]. 百度百科，2017-03-15 [2017-01]. <http://baike.baidu.com/item/github>
17. [suffersky](http://www.baidu.com/p/suffersky). Tomcat释义[CP/OL]. 百度百科，2016-09-23 [2017-01]. <http://baike.baidu.com/item/tomcat>
18. [风尘叹](http://www.baidu.com/p/%E9%A3%8E%E5%B0%98%E5%8F%B9). Mysql释义[CP/OL]. 百度百科，2017-3-20 [2017-01]. <http://baike.baidu.com/item/mysql/471251>
19. Java118. Spring释义[CP/OL]. 百度百科，2017-02-14 [2017-01]. <http://baike.baidu.com/item/spring/85061>
20. [weiyao\_85](http://www.baidu.com/p/weiyao_85). IoC释义[CP/OL]. 百度百科，2016-11-03 [2017-01]. <http://baike.baidu.com/item/控制反转/1158025>
21. [flyingship](http://www.baidu.com/p/flyingship). AOP释义[CP/OL]. 百度百科，2017-01-06 [2017-01]. <http://baike.baidu.com/item/AOP/1332219>
22. [jioluna](http://www.baidu.com/p/jioluna). ORM释义[CP/OL]. 百度百科，2017-04-19[2017-01]. <http://baike.baidu.com/item/ORM>
23. [wwcs](http://www.baidu.com/p/wwcs). Hibernate释义[CP/OL]. 百度百科，2017-04-27[2017-01]. <http://baike.baidu.com/item/Hibernate/206989>
24. [cnazhp](http://www.baidu.com/p/cnazhp). Sitemesh释义[CP/OL]. 百度百科，2016-08-29[2017-01]. <http://baike.baidu.com/item/sitemesh>

1. 使用过滤器解决sitemesh与FreeMarker结合装饰页不渲染的问题<http://www.cnblogs.com/adaikiss/archive/2012/05/08/2490339.html> [↑](#footnote-ref-1)
2. 这里感谢林禹森同学提出的建议 [↑](#footnote-ref-2)