# 项目总结报告

#### 项目简介

项目要求：本学期项目为小组自选项目，小组提出创意，并进行需求收集、整理，分析项目可行性。项目要求有创新性、实用性，并能综合应用所学的专业知识。项目采用敏捷开发的过程管理，并借助相应管理工具完成需求变化管理、版本控制、缺陷管理、配置管理等工作。

#### 团队成员及角色

#### 项目Product Backlog

#### 项目sprint计划

#### 项目的缺陷管理

5.1 Sprint1

A．配置文件，spring 配置，springMVC配置

解决方案：配置问题不是缺陷问题，错误的缺陷定义

5.2 Sprint2

B．服务器与浏览器之间的加密传输

解决方案：使用hash，加密传输的密码

5.3 Sprint3

A．当上传大文件时浏览器崩溃

B．上传时间较长

解决方案：尽量不上传大文件

5.4 Sprint4

A．网页加载缓慢，运行卡顿

解决方案：由于数据库中的内容不经常改变，可以将sql的结果缓存在服务器中，设置更新间隔，浏览器直接中缓存文件加载结果

#### 项目的配置管理

* 源代码的配置管理
* 文档的配置管理

6.1 源代码配置管理

6.1.1 项目GitHub地址

<https://github.com/ilerJemos/SECP_5C1K>

6.1.2 配置管理目的：

a. 解决多个用户对同一文件进行修改所引起的版本不一致问题；

b. 解决多个用户对同一个文件同时进行修改所引起的资源冲突问题；

c. 解决丢失版本或不知版本问题；

6.1.3 分支管理

a. master 主分支，发布项目release 版本

b. xpf 肖鹏飞

c. develop 孙林

d. HeisenbergYang-patch-1 杨歌

e. lxy\_branch 刘晓宇

6.1.4 文件管理

a. src 项目运行的后台

b. WebContent 项目前端界面

c. Doc 项目相关文档

d. Source 项目相关，数据库资源

6.2 文档配置管理

6.2.1 技术文档

用户需求说明书，概要设计文档，详细设计文档

6.2.2 项目管理文档

项目计划，项目配置管理计划

6.3 开发配置管理

开发工具：eclipse, Intellij, pycharm

开发语言：Java, Python

使用工具：maven

数据库：MySQL

硬件平台：全平台

浏览器：chrome, firefox , safari, ie9及以上

JDK版本：java 1.8

#### 项目架构设计

* 项目可能采用的架构，优缺点分析。
* 架构的决策（从质量属性方面分析）。a
* 架构的设计，画架构图。

7.1 项目采用的架构及优缺点分析

7.1.1 mybatis的优缺点：

优点：

A．sql写在xml文件中，便于统一管理和优化，解除sql和程序代码的耦合。

B. 供映射标签，支持对象和和数据库orm字段关系的映射，支持对象关系映射签，支持对象关系的组建

C. 提供xml标签，支持编写动态sql

D. 提供各种用途、功能的插件

缺点：

A．工作量较大，特别是在表的字段多，关联表多的情况下

B．sql语句的编写依赖于数据库，移植性差。

C．不支持级联删除，级联更新，需要自己对表进行删除。

7.1.2 spring的优缺点：

优点：

A．通过Spring的IOC特性，将对象之间的依赖关系交给了Spring控制，方便解耦，简化了开发。

B．通过Spring的AOP特性，很容易实现事务，日志，权限的控制。

C．提供了对其他优秀开源框架的集成支持。

D．通过pojo 进行轻量级和最小侵入式开发

缺点：

1. jsp中需要写大量代码，控制器过于灵活，缺少一个公用控制器
2. 不支持分布式

7.1.3 SpringMVC的优缺点：

优点：

A．springMVC是使用了MVC设计思想的轻量级web框架，对web层进行解耦，是的我们开发更简洁。

B．与Spring无缝衔接。

C．灵活的数据验证，格式化，数据绑定机制。

缺点：

1. 学习曲线比较陡峭
2. 需要花费时间部署

7.1.4 Maven的优缺点：

优点：

1. 对第三方依赖库进行统一的版本管理
2. 统一项目的目录结构
3. 统一软件构建阶段
4. 支持多种插件

缺点：

1. 需要配置环境变量
2. 资源库大多在国外，build项目速度缓慢
3. 第三方依赖库版本混乱，不完整

7.2 架构的决策

7.3 架构的设计

#### 项目中设计模式和设计原则的应用

* 项目中设计原则的应用
* 项目中设计模式的应用
* 相关设计（类图体现）

8.1 设计原则的应用

8.1.1 单一职责原则（Single Responsibility Principle）

对于一个类而言，应该仅有一个引起它变化的原因。

8.1.2 开放封闭原则（Open Closed Principle）

对扩展开放，对修改关闭。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。简言之，是为了使程序的扩展性好，易于维护和升级。

8.1.3 里氏替换原则（Liskov Substitution Principle）

里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。LSP 是继承复用的基石，只有当派生类可以替换掉基类，且软件单位的功能不受到影响时，基类才能真正被复用，而派生类也能够在基类的基础上增加新的行为。里氏代换原则是对开闭原则的补充。实现开闭原则的关键步骤就是抽象化，而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。

8.1.4 最少知识原则（Least Knowledge Principle）

一个实体应当尽量少地与其他实体之间发生相互作用，使得系统功能模块相对独立。

8.1.5 接口隔离原则（Interface Segregation Principle）

使用多个隔离的接口，比使用单个接口要好。降低类之间的耦合度。由此可见，其实设计模式就是从大型软件架构出发、便于升级和维护的软件设计思想，它强调降低依赖，降低耦合。

8.1.6 依赖倒置原则 （Dependency Inversion Principle）

针对接口编程，依赖于抽象而不依赖于具体。

DI 是Spring 的 主要设计原则

8.2 设计模式的应用

8.2.1 代理模式

mybatis框架调用mapper进行数据库操作时，获取的是mapper接口而不是实现类，用到的便是动态代理思想。一般实现动态代理的方法有JDK反射机制提供的代理、CGLIB代理等。

8.2.2 单例模式

框架中是由SqlSessionFactory创建SqlSession的，也即会话，类似JDBC中的Connection。在每次进行程序-数据库操作时都需要创建SqlSession，因此该SqlSessionFactory会工作在整个生命周期内，如果每次进行操作时都新建它时会极大地消耗连接资源，而且职责也唯一，因此可以采用单例模式生成的SqlSessionFactory来管理程序中的SqlSession。此外SqlSessionFactory由SqlSessionFactoryBuilder通过读取XML或者Java编码获得资源来构建。

8.2.3 工厂模式

定义一个用于创建对象的接口，让子类决定实例化哪一个类。Factory Pattern使一个类的实例化延迟到其子类。   
spring中的FactoryBean就是典型的工厂方法模式。

8.2.4 MVC模式

8.2.5 适配器模式

将一个类的接口转换成客户希望的另一个接口，使得原本由于接口不兼容而不能一起工作的类可以一起工作

8.2.6 观察者模式

当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新

8.2.7 策略模式

使得算法可独立与它的客户而变化。Spring 在实例化对象的时候用到Strategy模式

#### 项目总结（心得体会）

合理的分工和项目规划可以极大的提高项目开发的效率。

需求一直在变，需要不断的对项目进行更新。

### 项目提交物：

* 项目总结报告
* 源代码
* 可安装的执行版本
* 用户手册（系统安装说明、使用说明）