太原理工大学

基于SSM国家逃犯网络在线查询系统的  
设计与实现

软件工程学士学位毕业设计（论文）

刘磊

指导老师：张玲

2019年5月

TaiYuan University of Technology

Design And Implementation of Fugitive Criminal Query System   
Based on SSM

A Thesis Presented to the College of Software Engineering  
In Partial Fulfilment of the Requirements for  
The Degree of Bachelor of Engineering

By

Lei Liu

Under the Supervision of Lecturer Ling Zhang

May 2019

摘要

SSM 即Spring、SpringMVC、MyBatis，是Java平台的现代Web应用开发最成熟的程序开发框架组合之一。Spring即SpringCore提供了面向对象程序设计中通过依赖注入实现控制反转的基本核心库。SpringMVC则提供了对JavaEE平台上的Servlet技术的高层封装，提升开发效率；同时正如其名MVC(Model, View, Controller)， SpringMVC提供了基于模型-视图-控制器模式的web开发功能，实现了web应用开发过程中的架构分层，同时新版本SpringMVC也跟随现代web应用开发的最新技术，提供了对RESTful API开发的支持。MyBatis则是目前java平台流行的ORM（对象关系映射）框架，通过定制化SQL配置将POJO映射到数据库关系模式，且其提供的动态SQL可以非常灵活的根据参数状况改变SQL成分，解决了原生JDBC开发中拼接SQL遇到的格式化难的问题甚至根本不再需要直接接触任何JDBC代码。

本文通过基于上述技术设计和实现一个“逃犯在线查询系统”的过程来展现现代Java平台web应用开发的前沿技术，并通过实现一个单点登录模块来简要体现微服务架构的web应用设计，另外通过使用Kubernetes（Docker based）部署项目来介绍容器化应用部署对于微服务架构的重要性。

关键字：SSM，SpringBoot，微服务，容器化，查询系统

Abstract

目 录

[摘要 1](#_Toc7905235)

[Abstract 1](#_Toc7905236)

[1. 绪论 1](#_Toc7905237)

[1.1. 课题背景 1](#_Toc7905238)

[1.2. 本文研究内容 1](#_Toc7905239)

[1.2.1. 逃犯查询系统 1](#_Toc7905240)

[1.2.2. 现代web应用开发技术 1](#_Toc7905241)

[1.3. 本文结构 2](#_Toc7905242)

[2. 开发与运行环境介绍 3](#_Toc7905243)

[2.1. 开发环境 3](#_Toc7905244)

[2.1.1. 概述 3](#_Toc7905245)

[2.1.2. 硬件环境 3](#_Toc7905246)

[2.1.3. 软件环境 3](#_Toc7905247)

[2.2. 线上环境 3](#_Toc7905248)

[2.2.1. 概述 3](#_Toc7905249)

[2.2.2. 云计算资源详情 4](#_Toc7905250)

[2.3. 开发过程辅助工具 4](#_Toc7905251)

[2.3.1. 版本控制系统 – Version Control System 4](#_Toc7905252)

[2.3.2. 依赖管理工具 – Dependency Management Tool 6](#_Toc7905253)

[3. 需求分析 7](#_Toc7905254)

[3.1. 可行性分析 7](#_Toc7905255)

[3.2. 系统需求分析 7](#_Toc7905256)

[4. 系统设计 8](#_Toc7905257)

[4.1. 概要设计 8](#_Toc7905258)

[4.2. 数据库设计 8](#_Toc7905259)

[5. 系统实现 9](#_Toc7905260)

[5.1. 详细设计 9](#_Toc7905261)

[5.2. 关键算法 9](#_Toc7905262)

[6. 系统测试 10](#_Toc7905263)

[6.1. 测试任务及目标 10](#_Toc7905264)

[6.2. 测试方案 10](#_Toc7905265)

1. 绪论
   1. 课题背景

从1990年第一个网页浏览器诞生以来，web应用开发已经有了将近30个年头的历史，从最初的完全静态页面，到后来支持数据持久化和动态计算与显示。后来使用JavaEE平台来开发web应用成为最广泛的方案，并且在此过程中出现了非常经典的MVC应用架构。但是MVC应用架构是单体应用时代的产物，主要采用的是服务端对数据进行处理并直接填充进最终的html代码中最后返回给浏览器的内容是一切都组装好的。

最近的几年web应用的开发出现了更大的变化，MVC的方式正在逐渐成为过去，更新的方式是前后端完全分离。后端应用编写时不用再考虑视图的问题，只需要通过HTTP协议对外提供数据接口即可；使用JSON[[1]](#footnote-1)作为数据序列化格式，web API 采用REST[[2]](#footnote-2)风格设计，然后前端应用使用MVVM[[3]](#footnote-3)框架来构建单页面应用，全部的数据交互都是通过发起HTTP异步请求的方式。

* 1. 本文研究内容
     1. 逃犯查询系统

拿到题目后通过多种途径搜索资料关于逃犯查询系统的信息少之又少，几乎没有相关的文献或项目。对于逃犯查询系统的相关功能，在公开资料中本文没有可借鉴资源。所以对于功能需求，将仅实现基本的逃犯信息的增删查改。

* + 1. 现代web应用开发技术

本论文的研究重点一方面是设计实现逃犯查询系统，更重要的一方面是在这个过程中来尽可能多的使用web应用软件开发领域的最新技术来开发、架构和运行一套web应用软件并以此宏观的介绍web前端开发、web后端开发以及web应用实施部署方案的最新技术。主要包括web前端开发技术中的MVVM框架--Vue、微服务web应用架构、后端开发方案--RESTful API、使用Docker容器技术来部署各种服务以及使用Kubernetes在集群中进行容器编排与负载均衡。

* 1. 本文结构

本文主体是一个web应用软件的开发文档，在软件开发过程的每个阶段对应的文档的适当位置来叙述现代web应用开发使用的新技术。

1. 开发与运行环境介绍
   1. 开发环境
      1. 概述

在开发环境使用直接使用Docker来运行PostgreSQL和Redis等持久化和中间件工具，通过定义容器镜像文件来锁定各种软件版本；在线上环境部署应用时直接使用定义好的镜像和软件配置来创建容器实例。如此可消除由于开发环境与生产环境的软件差异带来的部署和调试困难的问题，另外也可以大大加快应用部署速度。

* + 1. 硬件环境

表 1 开发环境硬件环境

|  |  |
| --- | --- |
| 硬件名称 | 硬件信息 |
| CPU | Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz |
| 内存 | 16 GB 2133 MHz LPDDR3 |
| 显示 | 13.3－英寸 (2560 x 1600)  Intel Iris Plus Graphics 640 1536 MB 图形卡 |

* + 1. 软件环境

表 2开发环境软件环境

|  |  |  |
| --- | --- | --- |
| 用途 | 软件项目 | 版本 |
| 操作系统 | MacOS Mojave | 10.14.4 |
| 容器引擎 | Docker | CE 18.09.2 |
| 数据库管理系统 | PostgreSQL | 11.2 |
| 缓存数据库 | Redis | 5.0.4 |
| 程序平台 | Java (JDK \ JRE) | 11.0.2 |
| IDE | Intellij IDEA | Ultimate 2019.1 |

* 1. 线上环境
     1. 概述

本文所涉及的软件项目在演示阶段将部署于华为云计算平台，使用华为云提供的“云容器引擎CCE”通过Kubernetes来管理软件应用所有组件的发布和运行。使用Docker官方镜像仓库Docker Hub来分发设定好的容器镜像，再通过华为云CCE拉取镜像构建容器最终运行并对外。具体购买的华为云计算资源详情详见下表。

* + 1. 云计算资源详情

表 3云计算资源详情

|  |  |
| --- | --- |
| 项目名称 | 项目信息 |
| 弹性云服务器ECS | 通用计算型|s2.large.2  2核CPU，4GB RAM |
| 操作系统 | Linux 3.10.0 |
| 网络 | 静态BGP, 固定带宽 1Mbps |
| 云容器引擎CCE | Kubernetes 1.11.7-r0  Docker 17.06.0.60 |

* 1. 开发过程辅助工具
     1. 版本控制系统 – Version Control System
        1. 版本控制解决什么问题

软件开发过程通常是一个不断改进和迭代的过程，随着需求的不断完善和细化以及系统设计的改进和重构，软件代码需要不断的满足要求而发生变化。

对于代码量很小的项目可能由单个开发者凭借对项目的熟悉即可熟知每一个变更细节，掌控全局代码；但是对于一个代码量比较大，功能模块较多的复杂软件系统而言，代码的管理变得非常困难；在代码作出修改版本进行演进之后很可能会出现bug，这时无法追踪代码的改变会使得代码的修复变得比开发还困难。

另一方面，通常一个复杂的系统不是由单个程序员完成开发的，需要有一个团队甚至数个团队的通力配合来共同完成一套系统的代码实现。如何将所有成员的开发成果进行集成、运行、调试、测试也是一个非常重要的问题。

另一个问题是，对于现代软件工程项目，通常是线上运行着一个稳定版本，而开发团队在不断研发着好多新特性，也会有不同的小团队来负责不同的特性开发；这些由不同的小组负责的新功能特性开发工作常常是同步进行的。这样就可能导致为了不同的开发任务编辑或修改了软件产品的同一处代码实现从而产生了代码冲突，此时如果没有一个强大的工具来管理代码，似乎新的几个功能根本没办法同时集成进已有系统。

* + - 1. 版本控制系统如何解决问题

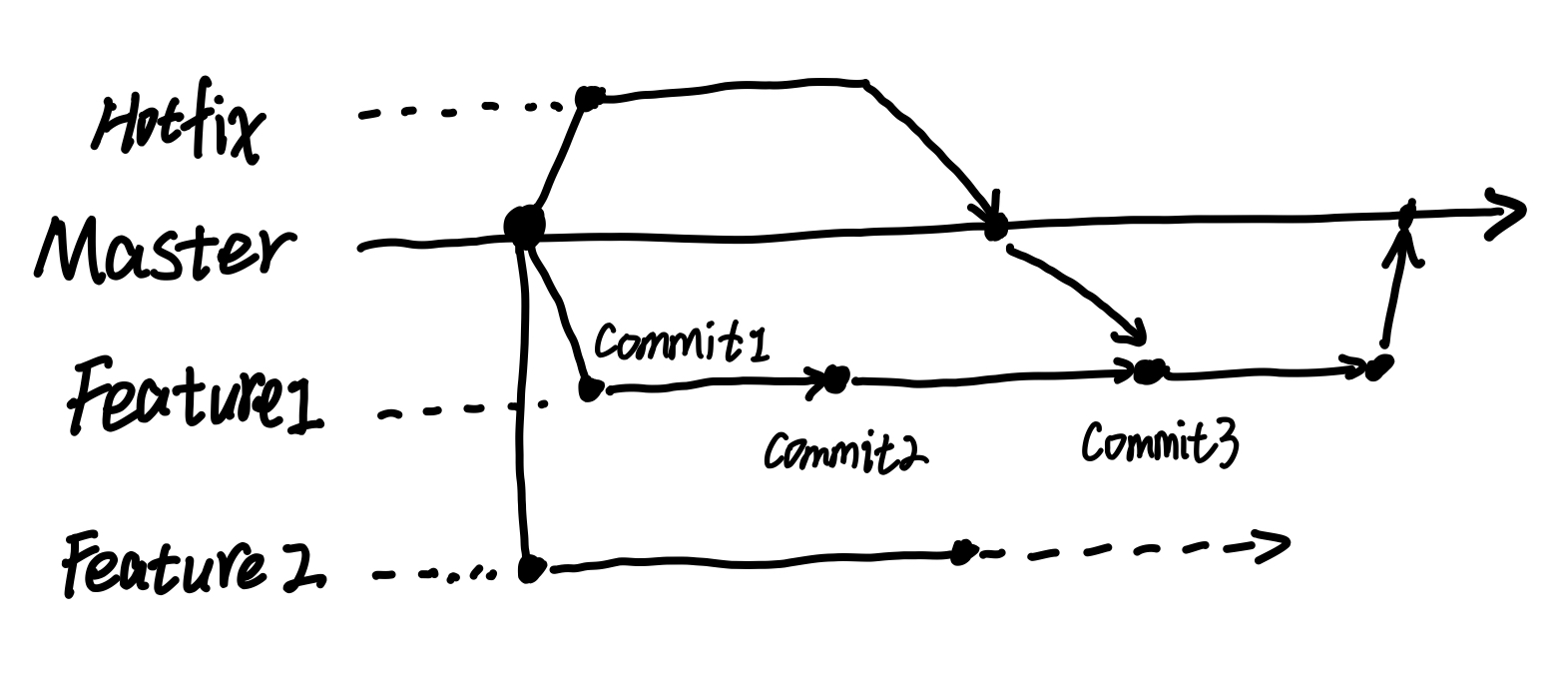
首先VCS的最基本任务就是记录代码的修改。使用版本控制系统来追踪(Track)源代码文件，进行过一些修改后通过提交命令(commit)来提交本次修改，这样版本控制系统就会记录下这次修改的内容；随后可能发生了很多次commit，版本控制系统都会记录下来。价值就在于对每一次commit，随时都可以找回(checkout)查看并重新编辑之前发布的内容，每一个commit记录都是“时光机”的跳跃点。

VCS的另一个重要作用就是解决线上线下多个软件版本在多个软件开发小组之间的协作问题。VCS通过“分支”(branch)机制来管理处在不同开发进度的多个软件版本；对于每一个正在研发的新功能特性都可以建立一个分支，这些分支和主版本之间的关系就像树枝与树干的关系，各自可以有自己的版本演进(commit)；而强大之处就在与这些分支可以随意的进行合并(merge)，只要能解决不同分支之间的冲突[[4]](#footnote-4)(conflict)即可。

对于团队协作，VCS发挥了不可忽略的作用，所有的开发人员不需要考虑使用什么方式将自己的代码共享给其他人，大家通过VCS就可以看到所有分支的所有commit状况[[5]](#footnote-5)。

另外VCS在现代web应用的维护中还担任着持续集成(CI: Continuous Integration)与持续交付(CD: Continuous Delivery)过程中的代码资源库的角色。对于其他VCS的作用将不再深入讨论。

图 1产品分支模型



* + - 1. VCS的选择

本文中的示例项目将使用目前最流行的版本控制工具GIT来进行版本控制，并使用GitHub来作为远程git仓库来保存代码；项目将在本论文通过所有审核后变为开放访问权限。

GitHub地址： 后端项目：<https://github.com/linkinghack/criminalquery>

前端项目：<https://github.com/linkinghack/CriminalQueryApp>

* + 1. 依赖管理工具 – Dependency Management Tool
       1. 依赖问题

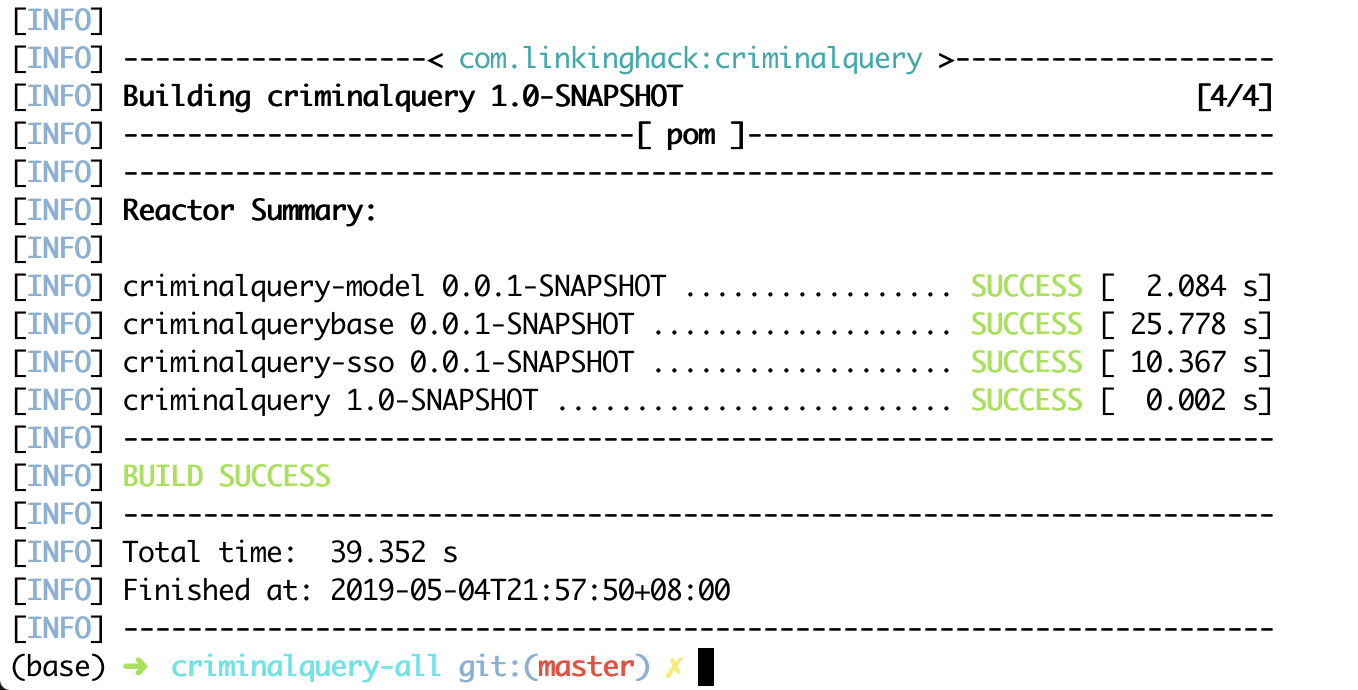
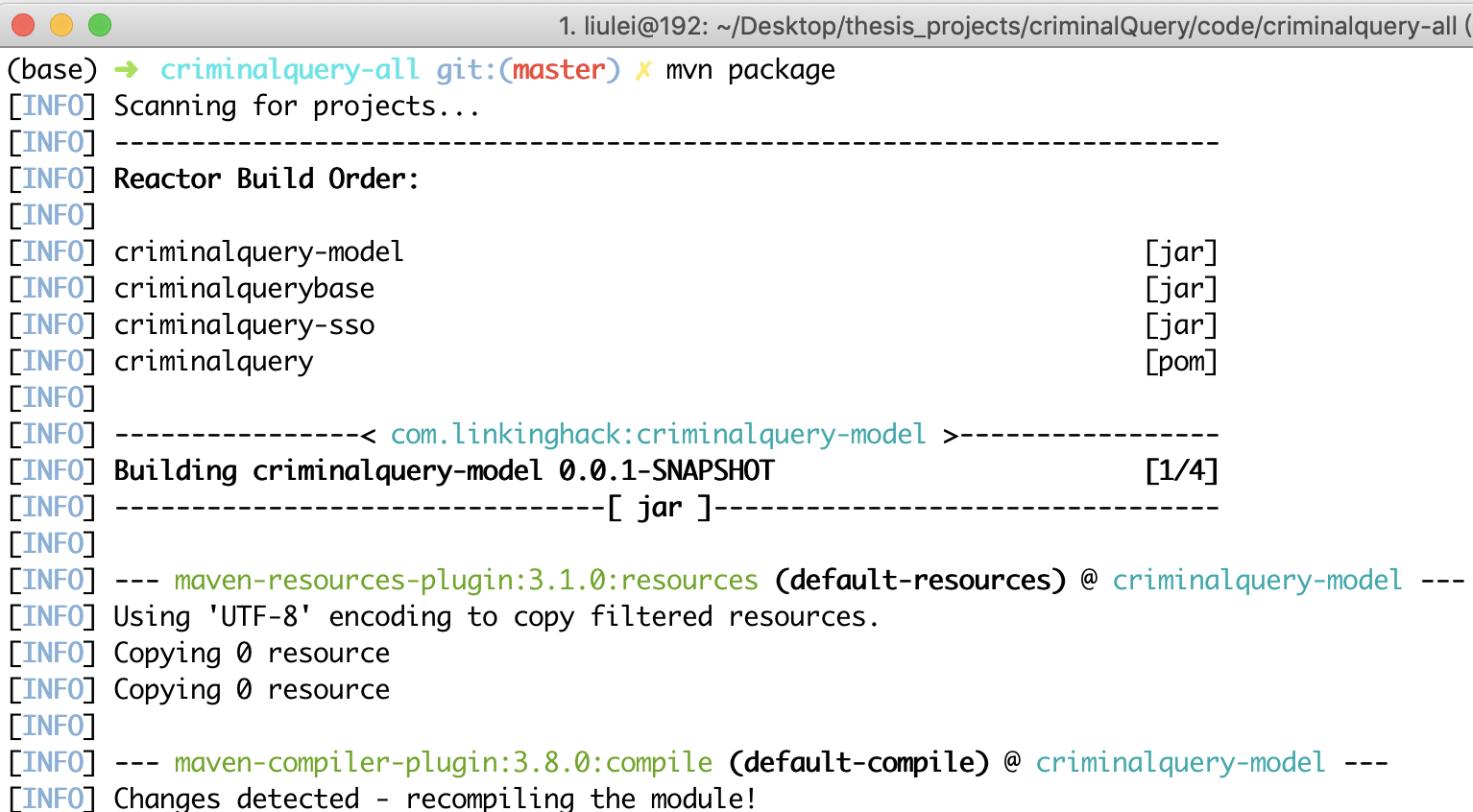
在软件开发过程中时常需要调用一些第三方库来利用开源成熟解决方案解决软件开发中的某些问题。事实上这是一个发生非常频繁的事件，并且依赖项目之间也有相互的依赖关系。

* + - 1. 解决方案

在本文所述的“逃犯查询系统”中，将使用Java平台比较流行的Apache Maven[[6]](#footnote-6)来管理依赖组件。

Maven可以用来管理Java软件项目开发过程中的依赖jar包，也可以用来进行软件构件(build)，还可以通过模块化的方式来拆分项目，并解决模块间的依赖调用；配合IDE的提示功能在模块间的依赖代码也可以获得完整的代码提示。

图 2 Maven打包命令示例



1. 需求分析
   1. 可行性分析
   2. 系统需求分析
2. 系统设计
   1. 概要设计
   2. 数据库设计
3. 系统实现
   1. 详细设计
   2. 关键算法
4. 系统测试
   1. 测试任务及目标
   2. 测试方案

1. JSON即JavaScript Object Notation，一种数据序列化格式，使用key:value的方式来嵌套的定义数据结构 [↑](#footnote-ref-1)
2. 即Representational State Transfer，详见https://zh.wikipedia.org/wiki/表现层状态转换 [↑](#footnote-ref-2)
3. MVVM即Model-View-viewmodel, 详见https://zh.wikipedia.org/wiki/MVVM [↑](#footnote-ref-3)
4. 冲突即不同的分支修改了同一个文本文件的统一行内容，VCS不知道应该保留那一个分支的修改，此时需要认为决定。 [↑](#footnote-ref-4)
5. 这里的“所有”指的是权限范围内的所有，VCS的商业版本（比如gitlab）有比较完善的权限系统，可以控制代码可见的范围。 [↑](#footnote-ref-5)
6. <https://maven.apache.org/> [↑](#footnote-ref-6)