目录

[摘要 2](#_Toc10354600)

[Abstract 3](#_Toc10354601)

[第一章 绪论 4](#_Toc10354602)

[1.1 研究背景与意义 4](#_Toc10354603)

[1.2 持续集成国内外研究现状 5](#_Toc10354604)

[1.2.1 持续集成国内外研究现状 5](#_Toc10354605)

[1.3 主要研究内容 7](#_Toc10354606)

[1.4 论文结构安排 7](#_Toc10354607)

[第二章 持续集成相关理论分析 8](#_Toc10354608)

[2.1 软件集成 8](#_Toc10354609)

[2.2 经典软件开发模型的集成方式 9](#_Toc10354610)

[2.2.1 “Big-Bang”集成方式 9](#_Toc10354611)

[2.2.2 “迭代递增”集成方式 11](#_Toc10354612)

[2.2.3 “每日构建”集成方式 13](#_Toc10354613)

[2.2.4 持续集成 13](#_Toc10354614)

[2.3 本章小结 15](#_Toc10354615)

[第三章 持续集成方案设计和系统搭建 16](#_Toc10354616)

[3.1 持续集成系统组成 16](#_Toc10354617)

[3.2 持续集成运行流程 16](#_Toc10354618)

[3.3 Jenkins持续集成系统搭建 17](#_Toc10354619)

[3.4 本章小结 19](#_Toc10354620)

[第四章 基于持续集成在VR领域的实践 20](#_Toc10354621)

[4.1 VR生产环境的自动化配置脚本设计 20](#_Toc10354622)

[4.1.1 设计需求与方案 20](#_Toc10354623)

[4.1.2 设计过程 21](#_Toc10354624)

[4.1.3 结果分析 26](#_Toc10354625)

[4.1.4 总结 30](#_Toc10354626)

[4.2 VR应用测试情况自动报表脚本设计 31](#_Toc10354627)

[4.2.1 设计需求与方案 31](#_Toc10354628)

[4.2.2 设计过程 33](#_Toc10354629)

[4.2.3 结果分析 43](#_Toc10354630)

[4.2.4 总结 46](#_Toc10354631)

[4.3 本章小结 48](#_Toc10354632)

[第五章 总结与展望 49](#_Toc10354633)

[致谢 50](#_Toc10354634)

[参考文献 51](#_Toc10354635)

# 摘要

作为新兴的产业，虚拟现实的应用开发目前仍在不断探索当中。为了应对快速发展的虚拟现实应用，本人实习公司采用了敏捷开发的方式进行快速迭代开发，并使用Jenkins作为持续集成系统进行构建和测试。然而在虚拟现实设备生产和应用测试中，环境的配置和测试数据统计仍然是通过人工实现的，不适应于持续集成所提倡的自动化流程。

本文首先对软件集成进行原理分析，描述了集成的概念，进而引出各种经典集成方式。通过分析各集成方式的原理和缺陷，从最基础讲起，重点分析了持续集成的发展情况和具体要求。再通过设计持续集成系统，搭建了Jenkins持续集成服务器。

本文针对公司的需求，分析和研究了生产环境数据库自动化脚本的编写，和代表持续集成反馈机制的虚拟现实测试情况自动报表的脚本两个实践，同时该两个实践都运行在Jenkins服务器环境中。

**关键词：**Jenkins，持续集成，自动化测试，虚拟现实

# 

# Abstract

As an emerging industry, the application development of virtual reality is still being explored. In response to the rapidly evolving virtual reality application, my internship company used agile development for rapid iterative development and built and tested using Jenkins as a continuous integration system. However, in virtual reality device production and application testing, environment configuration and test data statistics are still implemented manually, and are not suitable for the automated processes advocated by continuous integration.

This paper first analyzes the principle of software integration, describes the concept of integration, and then introduces various classic integration methods. By analyzing the principles and shortcomings of each integration method, we focus on the development and specific requirements of continuous integration from the most basic. The Jenkins Continuous Integration Server was built by designing a continuous integration system.

In this paper, based on the needs of the company, the two practices of the production environment database automation script writing and the virtual reality test automatic report script are analyzed and studied, and both practices are run in the Jenkins server environment.

**Key words:** Jenkins, continuous integration, automated testing, virtual reality.

# 绪论

## 研究背景与意义

随着计算机软硬件技术的不断升级，以及互联网的快速发展，软件技术也进行着不断变换，大量软件不断地开发、迭代，软件的开发周期越来越短。自从虚拟现实（Virtual Reality，VR）技术发展以来，基于VR的应用也不断涌现出来。经过了数年的发展，VR的主流从当初的VR盒子（基于手机）到PCVR（基于计算机），再到一体机VR，VR应用也从移动VR应用发展到PCVR应用，再发展到移动VR应用。与VR盒子应用类似的是，一体机VR应用是基于Android系统的，因此大多数手机应用是可以在VR一体机中使用的；不同的是，VR一体机的UI系统是一个虚拟全景系统，而其应用多数是需要支持射线、定位、触控等功能的。得益于如今VR芯片的发展，一体机VR的性能也越来越高，结合各大VR厂商的软件开发工具包（Software Development Kit，SDK），基于一体机VR的应用必将成为VR的主流。

传统的软件构建方法是书序生命周期“瀑布”开发模型，通常都是开发初期制定详细的计划，在计划期间仔细研究、设计最终产品，并且将一切详细资料记录在案。这种方法比较有逻辑性，但是大大增加了开发周期，很难适应如今快速变化的软件开发，无法保证快速开发中的软件质量。这种情况下敏捷开发就诞生了，而持续集成（Continuous Integration，CI）作为敏捷开发的最佳实践被大量开发团队使用，引入自动化测试代替手工测试，测试驱动开发（Test-Driven Development，TDD）成为支撑短周期快速迭代的有效方法。

然而，在一体机VR成为主流前，多数VR厂商将重心都放在了硬件研发和理论研究上，VR应用研发者们只需要考虑Windows系统和显卡的兼容性，之前PC的VR应用研发也都集中在大型的游戏上，开发周期都比较长，“瀑布”开发模型完全可以支撑。对于一体机VR，VR厂商们需要开发自己的VR系统和SDK，各种应用也转变为轻量级应用，开发者们同时需要考虑各种硬件、系统的兼容性问题，敏捷开发成为必然趋势。然而这种转变是很困难的。以本人所实习公司为例，拘泥于传统的开发模式，虽然公司使用了Jenkins对应用自动编译部署并发布，但是作为持续集成的核心，自动化测试并没有落实，研发习惯了简单粗暴的功能编写，而测试则主攻在手动测试上，再加上VR技术的快速发展，自动化测试初期的成本太大，公司只能招入大量的测试人员进行手动测试，增加了人力成本。人员的流动性太大，往往会导致业务了解上的断层，测试者们花大量时间在业务的沟通了解上，过时的测试用例不断累积，新的测试用例却很少创建。对于研发而言，由于没有测试驱动，往往在新功能添加后会引发出旧的功能出现问题，这又增加了测试成本，甚至会造成整个功能块的奔溃。

综合上述问题，需要设计一套完整的自动化测试框架支撑的持续集成方案，一方面解放大量的人工测试，另一方面研发者们在编写功能模块式可以自动测试防止旧功能出现问题。

## 持续集成国内外研究现状

持续集成发展到现在，主要是以美国微软和谷歌等公司带头发展，从概念的提出以到开源工具的使用，国外开发者社区都比国内相对活跃，所以研究国外持续集成的发展状况，并对比国内发展现状以及公司目前使用持续集成的情况是非常必要的。而作为持续集成的核心部分，自动化测试也需要有所体现。由此，本章将对持续集成和自动化测试的发展现状进行研究。

### 持续集成国内外研究现状

自上世纪九十年代以来，软件开发大师Kent Beck一直致力于研究高效的软件开发模式，1996年，他通过编写《Extrme Programming Explained: Embrace Change》一书提出了极限编程（ExtremeProgramming，XP）的概念，该书总结了XP的12个最佳实践，持集成这个概念最初就来源于其中一个最佳实践。然而持续集成实践在当时被当做是XP的一部分，由于XP中的许多实践不为业界所接受，持续集成也未能很好的发展。

2000年，同为XP思想发起者之一的软件大师Martin Fowler专门以《Continuous Integration》（持续集成）为题发布了一篇著名的文章，该文章以他所在公司ThoughtWorks的项目持续集成实践为基础介绍了持续集成的价值。他认为持续集成并不受限于XP的开发模式，其价值在于让项目团队高效地开发出一致的软件。持续集成逐渐得到人们的认可，但是在实际项目中应用还需要大量的研究和时间。

2006年3月，Martin Fowler正式对持续集成实践作出了描述，这里结合个人理解对其进行描述：

（1）维护源码仓库。在软件工程当中，存在大量的源码和其他依赖文件，由于项目修改编译的频率越来越高，大量的文件需要可以对其历史记录和相应的修改者进行回溯，保证在产生错误后能够迅速地上报相应的错误代码和相关文件。由此，需要相应的工具作为支撑，如配置管理工具、源码管理工具、版本控制工具、仓库工具。

（2）自动化构建。项目构建即将代码编译或解释成一个应用程序，整个过程需要对文件、数据库等等进行操作，这些操作往往是复杂且固定的，由此可以使用将这些操作进行自动化。如Java的自动化构建工具Maven和Linux中的自动化构建工具Make等等都可以实现类似的操作。

（3）令构建自测。每次构建完成后，为了保证构建是成功的，需要对构建完成的应用进行基础测试，自动化测试在这时是十分重要的。自测后保证构建好的应用可以正常运行，再交付给相关测试人员进行具体的功能测试。

（4）每人每天向主干提交代码。集成的目的之一是项目沟通，通过集成，项目开发人员在每次提交代码后会产生相应的代码修改记录，其他项目人员可以通过查看代码记录找到相应的负责人员，由此更好地进行代码和功能的交流。当项目存在错误时，尽早、少量的提交变更可以更快地定位到产生错误的变更，降低代码修复或回滚的风险和成本。

（5）每次代码提交前需在集成环境中进行一次构建。每次提交后构建再结合自测可以保证每次的提交是有效的，如无效则自动回退之前的版本。

随着持续集成的不断实践，越来越多的研发者们开始使用持续集成的模式，并基于持续集成开发了各种工具，如CruiseControl、Hudson、LuntBuild、Bamboo等等。如今，开源持续集成工具Jenkins（原Hudson）因其免费、跨平台、丰富的插件成为最受欢迎持续集成工具。在持续集成之后，持续交付和持续部署的概念也逐渐被提出来，Jenkins2.0也由此诞生，其精髓为Pipeline as Code，即任何发布流程都可以表述为一段脚本，实现由持续集成到持续交付的转变。

作为软件开发巨头，Google的持续集成技术是比较领先的。《How Google Tests Software》（Google软件测试之道）一书中详细描述了Google在如今快速变化的互联网时代，为了保证软件质量所做的自动化测试与持续集成的研究与发展，更加证实了持续集成在快速开发中保证质量的重要地位。

2013年，Paas服务提供商dotCloud发布了一个开源项目Docker，它是一个基于LXC容器技术的高级容器引擎，之后大量开发者加入到了这个开源项目当中，开发者们在不断实践中将Docker应用到持续集成当中，使持续集成环境变得更加易于部署和移植。之后，大量的Docker镜像可以被使用，其中就包含Jenkins持续集成环境。

目前国内各大厂都有各自的持续集成方案，阿里、腾讯、华为这几家云服务也引入了Docker技术，华为工程师们也在《Docker进阶与实战》中描述了一些基于Docker的持续集成实践

## 主要研究内容

（1）针对一般软件开发过程的继承问题，分析传统软件开发模式集成方式的不足，引入持续集成思想，对持续集成进行深入的分析与研究。

（2）基于持续集成的理论，搭建Jenkins持续集成服务器。

（3）研究在VR生产环境中自动配置数据库的脚本。

（4）研究公司VR测试情况的自动报表脚本。

## 论文结构安排

第一章是绪论

第二章对持续集成相关理论进行分析

第三章介绍基于Jenkins的持续集成方案

第四章研究两个基于持续集成在VR领域的实践

第五章对持续集成在VR领域的总结与展望

# 持续集成相关理论分析

## 软件集成

在人类社会发展以来，集成始终贯穿着人类的各个领域，从社会的阶层统治到制造业中的生产流水线，一个庞大的产物总是由各个部分分工合作组成的，不同的领域也赋予了集成不同的定义，但是他们本质上都是一样的，这种模式是目前人类发展至今最有效的组成方式了。它保证了内部的稳定协调，也保证了外部的完整和稳固。

如今已经发展到了信息化社会，不同于传统的工业工程，软件工程是构建于虚拟的，我们可以把它看成虚无缥缈的，但也确实存在的，它由信息构成。当然由信息构成的虚拟的东西并不是只存在于现在，过去的精神文化如小说等就是如此，人们在创作时虽然说是天马行空的，但是一个好的作品是在一个完整的流程创作出来的，它们都少不了构造的过程，我们也可以把它理解成为一个虚拟的工程构造。软件开发也是如此，但是得益于信息的无处不在和信息社会的迅速发展，它们比之传统和精神的工程有了更高的要求和更快的发展，它们的构造流程也更加严格，软件集成由此发展而来。

一个工程的完整和有效是评估工程质量的重要标准，软件集成保证了工程内部的完整统一，也实现了交付前的质量测试。简单的软件可能只是实现需求的功能，而一个软件工程包含了必要的功能代码、调试代码和测试代码，它们通过模块化构成了一个完整的软件产品，这就是软件集成。

但是在软件开发行业发展初期，由于模块众多，只要其中一个模块出现错误，那么整个软件工程就无法正常运行，并且定位出现错误的模块也是非常耗时耗力的，即使问题已经发现，但是如何去修改该模块而不会导致其他功能出现问题也是需要思考的。所以，出奇的软件集成是是非常容易出现问题的，从问题来源来看，主要有以下几类：

（1）模块功能问题。基础模块开发是由开发人员编写的，编写过程难免会产生一些疏忽，从而引发一些Bug，这些Bug可能只是一些简单的语法或格式错误，也可能是由于开发人员对于功能理解错误引起的功能缺失或错误。当然在修复已有缺陷时仍然可能引入新的Bug，对于人来说，这些都是合情合理、无可避免的。

（2）集成时的模块组装问题。在开发单一模块时往往很难发现它与其他模块一起构建时的问题，即使它本身没有任何问题。这些问题可能是由于模块接口产生的，也可能是集成本身的结构存在问题。

（3）合并集成问题。模块开发往往是并行的，当多个模块开发完成后再将其合并，合并前产品本来是没有任何问题的，各个模块的改动也是没有问题的，但是合并后各个改动可能会产生冲突，从而引发一些问题。

（4）操作和设置错误。如不同的开发人员使用的开发环境或版本不统一引发的问题，不同的测试人员使用的测试环境同导致测试结果不一。再如在提交新创建的源文件时未将其添加到版本控制管理工具当中，其他开发人员无法拉取该源文件，导致运行失败。这些错误主要还是体现在开发沟通上。

开发人员们在不断实践中，发现并试图解决这些问题，软件集成方式也在不断改进。软件集成方式往往和软件开发模型密不可分，在软件开发当中存在几个经典的软件开发模型，它们的集成方法大同小异，对比其中各个集成方式的缺陷和优点，我们可以试图得出持续集成是如何发展的。

## 经典软件开发模型的集成方式

### “Big-Bang”集成方式

在软件开发行业发展过程中，为了适应不断增长的软件需求，人们在不断实践中总结出了一些软件生命周期模型，这些模型在软件开发过程起到了很大作用，即使它们存在一些问题。

1970年，Winston Royce在论文《管理大型软件系统开发》（Managing the Development of Larger Software Systems）中，基于软件生命周期提出了著名的“瀑布模型”（water-fall model）。该模型将软件生命周期按固定顺序分为六个基本工程活动，相邻的活动之接是双向的，它们自上而下相互衔接，形如瀑布逐级而下，如图所示。

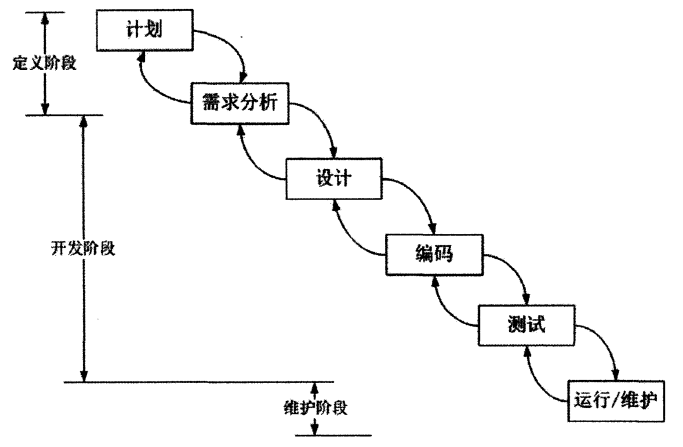


图2-1 瀑布模型

如图，瀑布模型将一个软件生命周期整体上分为三个阶段：定义阶段，开发阶段和维护阶段。定义阶段包含计划和需求分析工作；开发阶段包含设计、编码个测试工作；维护阶段包含运维工作。对于相邻的工作，它们之接的关系并不是单向的，下一阶段需要随时向上一阶段对产生的相关问题进行反馈，以支持项目某一阶段的更改。

根据瀑布模型的相关理论，一种叫“Big Bang”集成的软件集成方式应运而生。从瀑布模型来看，相邻的两个阶段是相关的，且后一阶段的启动必须依赖于前一阶段的产生，但是不相邻的两个阶段直接并未有所关联，这就造成在前一阶段进行规划时无法兼顾其他阶段，当所有阶段集成到一起时，某一阶段可能不符合预期或产生错误。而当对该阶段重新规划时，只能一级一级地向上反馈。由此可以看出，“Big Bang”集成方式意在当项目的进行到最后一个模块时再进行集成，当前面几个模块产生错误时只有在最后集成时发现，需要在最后集成时进行大量修改。在开发初期可能这种集成方式不会出现问题，但是到了软件快要要付时如果出现问题，那么无法很好精确预估修复时间，且各模块在此时已经趋于复杂化，严重地话可能需要对大量模块进行修改，增加时间成本，由此这种集成方式无法满足越来越复杂的软件工程了。

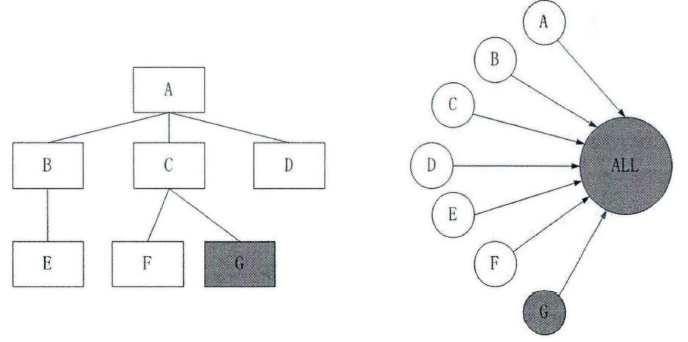


图2-2 Big-Bang 集成方式

### “迭代递增”集成方式

当软件需求在不断改变时，传统的“瀑布模型”无法有效快速地根据需求对软件工程进行修改。由于软件生命周期在不断变化，需要将整个周期按照某种方式划分为几个周期。“迭代递增”就是基于这种情况诞生的，它在Rational公司的Rational统一过程（Rational Unified Process，简称RUP）中提及。

“迭代递增”即存在迭代和递增两个主要方法，每一个迭代都是一个迭代周期，下次迭代都是在上一次迭代趋于稳定后逐渐复杂化，即称为递增。单个迭代周期可以看做是一个瀑布模型，它基本包含了瀑布模型的各个工作，如需求分析、设计、编码、测试工作。当一个迭代趋于结束后，首先要对其进行整体评估分析，确认该迭代周期已达到目标预期，可以结束。之后根据新的客户需求，对下一个迭代周期进行新的目标制定，做整体的规划调整。如图所示。

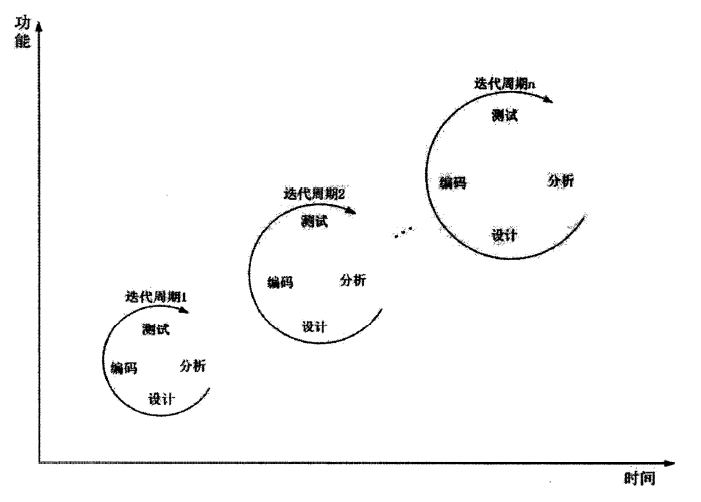


图2-3 迭代递增过程模型

虽然“迭代递增”解决了在客户需求不断改变下的周期问题，但是这种集成方法在不断实践下暴露出两个主要问题，即“迭而不增”和“增而不迭”。

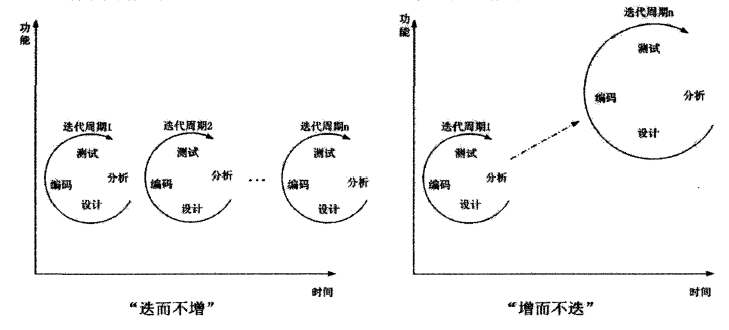


图2-4 迭代递增集成方式的两个问题

如图所示，“迭代递增”主要有两个需要量化的量，即迭代间隔和任务量，如何对它们进行限定成为该集成方式的一个主要问题。当迭代间隔较短时，两个迭代周期的任务量可能不会发生较大改变，仅仅是重复迭代；当迭代间隔较长时，两个迭代周期的任务量差距可能非常大，此时的集成方式其实回归到了“Big Bang”集成。微软公司提倡的“每日构建”集成方式很好的解决了这两个问题。

### “每日构建”集成方式

作为一家世界顶级的软件开发公司，微软无疑拥有者丰富的软件开发经验，他们的工程师在不断实践中根据经验总结出了很多软件开发的方法，其中就包含一套微软解决方案框架（Microsoft Solution Framework，简称MSF），其中就提到了基于里程碑的项目生命周期管理框架。

MSF过程模型中提到了著名的“每日构建”（Daily Build）软件集成方式。这种集成方式将迭代间隔固定唯一天，将迭代增量固定为一天的增量。即在项目组每天完成代码提交后，需要立即进行项目构建，并对构建好的程序进行可执行测试，这种测试被称作“生成验证测试”（Build Verification Test，简称BVT）。为了实现每日构建，微软项目组在项目开除其会通过需求分析，创建一个用于编译、生成、测试的框架，该框架可以实现对所有开发人员每日提交的代码进行合并、构建，最后再进行一次BVT测试。当测试中发现问题后，在下一次迭代周期进行修改，如此往复，保证了软件项目的持续稳定。

### 持续集成

互联网行业的迅速发展使得软件行业的规模变得越来越大，也越来越复杂，各种大小型软件开发相关企业不断出现，不同规模的公司也对持续集成有了不同的要求。由于软件行业的用户量及其庞大，各种需求都是迅速变化且不可预测的，为了应对种种情况，如今的敏捷开发方法被各公司所推崇。本人所在公司即使用这种开发方式。

敏捷开发推崇缩短软件版本开发的周期，简化开发流程，避免多次重复迭代引起的成本问题，持续集成能够很好的应对这种开发方式。持续集成更加注重自动化流程，以保证在一天之内可以有序无误且快速地构建并测试多个软件版本。对比前面的“每日构建”，持续集成并不局限于每天一次构建，“每日”仅仅是它的最低要求。一个完全自动化的持续集成流程包含自动化的构建和自动化测试以及其他项目管理流程，它可以实现每天不止一次的重复构建。从表面上看，虽然流程实现了自动化，多次构建仍然可能加大了开销，但是从长远来看，这种做法其实是节约成本的，并且保证了软件的持续稳定性。

开发人员通过不断实践发现，集成的工作量并不是累加的，而是与两次集成时间间隔的平方近似成正相关。我们假设每天集成一次需要的工作量为C，那么一周（五天）如果采用每天集成一周集成的工作量为5C，而一周集成一次的工作量为25C。从集成工作量来看，缩短集成间隔能够有效地减少总体集成的工作量。当然，对于工作量的计算是要考虑到项目计划、开发人员的感受等等，恰当的集成间隔会使开发人员感到舒适，且不会影响到整个项目的进程。

Martin Fowler提倡在任何可执行的软件开发过程中使用持续集成，在项目不断自动构建的同时保证测试数据的完整，也保证项目是可以溯源的，提高在每次修改的可行性，实现这些目标的持续集成需要包含以下几个环节：

（1）代码管理。由于一个项目的开发人员并不只有一个，为了实现项目的统一管理，且所有参与人员都可以获取、修改、运行他人的提交。当代码未进行统一管理时，开发人员需要随时进行代码的沟通，防止双方所开发部分不会产生冲突，这就需要大量花费时间去进行沟通。所以，首先需要一款代码管理工具将所有参与者的开发代码统一管理在一个地方，如服务器等，这样参与开发人员可以随时查看其他人所提交的代码；当然，代码的提交也要考虑其可行性，此时需要具有相关权限的开发人员对提交的代码进行评审，再决定该部分代码是否能合并到整个代码库中；除此之外，代码库中应不仅仅含有当前版本的代码文件，需要对项目整个过程的大多数版本进行备份，保证更改错误后可以快速回溯代码；当然，为了能保证代码库中的代码在所有参与人员的主机上是可以执行的，需要对整个项目的所有依赖进行同步，可以通过一些配置文件达到此目的。

（2）自动化构建。自动化构建要求项目代码能够通过简单的命令将其自动转化为可执行的应用程序。对于小规模的工程，他的源文件和依赖较少，几条命令即可完成构建；但是对于大型企业项目，源文件构造十分复杂，整体成树状结构，开发人员需要通过文件检索去寻找特定的文件，显然一个一个地对文件进行构建是不切实际的。那么就需要一款自动化构建工具，它能够实现将所有源文件和依赖按照一定的编译顺序，自动有序地编译出来，并且将相应的编译信息输出以供分析。如Java的自动化构建工具Maven和Linux中的自动化构建工具Make等等都可以实现类似的操作。

（3）自动化测试。自动化测试是极限编程（XP）中所提到的实践之一。XP中将测试分为两类：功能测试和单元测试。其中单元测试是由软件开发人员编写的用来测试代码的基本编写单元，这种测试基本都是基于脚本的，基本上都可以实现自动化测试；功能测试的范围较大，多数功能测试都是从客户需求和产品设计上考虑的，且都是基于用例驱动的。功能测试大多数都是依赖手动测试的，因为很多和功能设计需要从用户角度去考虑，这些都是感官上体现的。测试人员在不断实践中发现，有大量的功能测试在每个版本的操作都是重复的，是可以通过脚本自动化的。大多数这些功能都是项目中已经稳定不变的。每次构建完成后，为了保证构建是成功的，需要对构建完成的应用进行这些基础测试，自动化测试在这时是十分重要的。自测后保证构建好的应用可以正常运行，再交付给相关测试人员进行其他无法自动化的功能测试。

（4）主创建。开发人员在提交代码前需要要确保该代码是可以在整个项目中构建的，这就要求开发人员每天在提交代码前将代码服务器中的代码更新到本地，在进行本地构建。然而对于大型的项目，这种将所有代码都更新的方式无疑是浪费时间的，代码的保密性也不是很好。主创建提倡开发人员将自己所负责的模块代码在每次开发前同步到本地，在本地修改后提交到代码库中，并在服务器上进行构建。如果构建成功，那么代码提交是成功的。如果构建失败，代码提交失败，代码库自动回退提交前的状态。为了防止构建冲突，主创建必须保证每个开发人员只能在不同时间进行构建，即同一时间只会给一位开发人员发放构建令牌。

## 本章小结

本章首先对软件集成进行原理分析，描述了集成的概念，进而引出各种经典集成方式。通过分析各集成方式的原理和缺陷，从最基础讲起，重点分析了持续集成的发展情况和具体要求。

# 持续集成方案设计和系统搭建

## 持续集成系统组成

持续集成系统主要由版本控制库、持续集成服务器、构建工具、反馈机制和开发人员构成，他们相互配合并相互依赖，最终完成从开发版本到自动化构建，然后进行自动化测试，最后返回反馈结果。各个环节主要描述如下：

（1）开发人员。开发人员负责代码的最初生成，在开发人员进行需求确认和框架设计后，他会在本地进行一次构建，构建成功后会通过版本控制工具将代码提交到版本控制库中。

（2）版本控制库。版本控制库通过能够通过一个代码管理工具来管理项目的源码和其他依赖资源，所有的源码都集中在存储在代码管理库中。这样版本控制库能够根据版本提交的历史来管理各个版本的代码，为所有开发人员提供统一的源代码版本和环境配置。相关工具如SVN代码管理服务器，Git代码管理工具和Gitlab代码管理服务器等等。

（3）构建工具。构建工具常常用来编译、解释、执行脚本等等。常用的构建工具有Maven、Ant、Make等。构建工具可以通过简单的命令按照一定的流程自动化地执行构建过程。

（4）反馈机制。持续集成的一个很大优势是能够将构建结果进行反馈，及时提醒开发者应对反馈结果。一般来说反馈方式使用的是日志、邮件、报表等等，当构建过程出现问题时，持续集成工具能够及时地通过这些方式通知开发者，开发者可以对详细的反馈信息进行分析，在对项目进行调试和开发，从而及时地解决问题，保证构建的持续稳定。

（5）持续集成服务器。持续集成服务器使用持续集成工具搭建在一个独立的环境中，它可以结合其他项目管理工具，通过配置实现持续集成的整个自动化流程。持续集成服务器类似一个定时装置，可以对需要构建的项目进行定时构建、测试、反馈等。一般来说，目前使用广泛的是Jenkins持续集成工具。

## 持续集成运行流程

持续集成的整个流程的开端是开发人员，通过反馈机制构成一个闭环。在一名开发人员完成自己所属模块的开发工作后，首先需要在本地执行一次构建，构建成功后，将完成的代码通过版本控制公会局提交到代码管理库中。在一个分工明确的项目团队中，开发人员根据自己所负责的模块承担着不同的任务，他们都可以提交自己的变更，通过审批后即可上传到代码库中。开发人员在每次提交变更后都会触发持续集成的整个自动化流程。持续集成的具体过程如下：

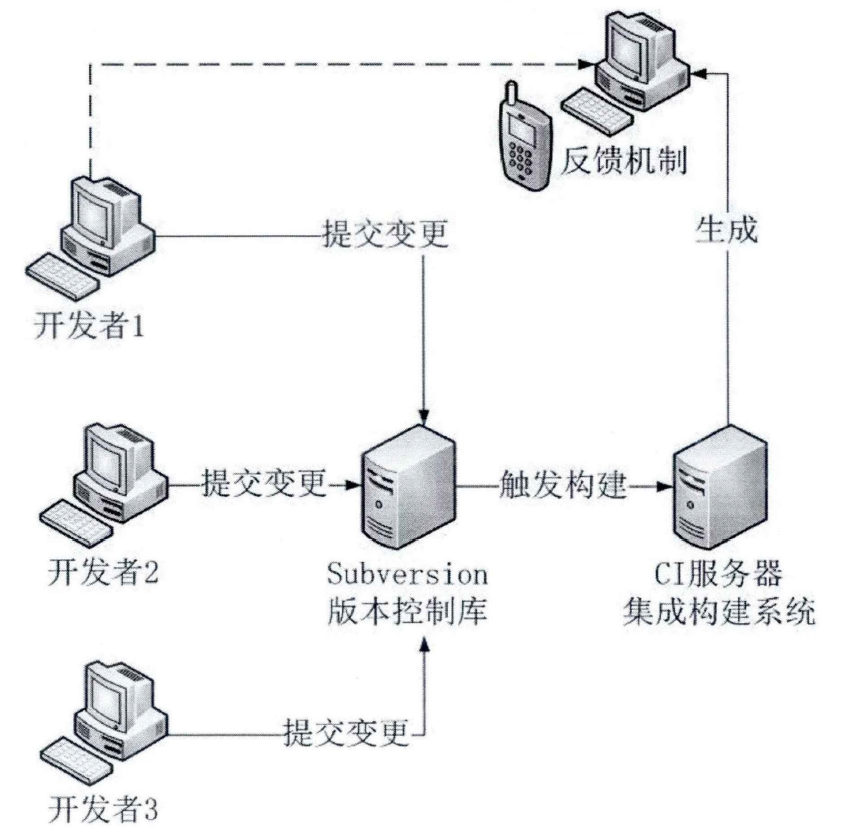


图 3-1 持续集成系统组成部分和运作流程

## Jenkins持续集成系统搭建

1. 使用VMware虚拟机安装Linux发行版Deepin。
2. 安装Docker

（1）首先添加使用HTTPS传输的软件包以及CA证书

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl python-software-properties software-properties-common

（2）添加软件源的GPG密钥

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add –

（3）添加docker软件源

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian jessie stable"

（4）安装docker-ce（docker社区版）

sudo apt-get update

sudo apt-get install docker-ce

1. 安装Jenkisn

（1）使用docker安装Jenkins

docker run \

-d \

-p 8080:8080 \

-p 50000:50000 \

-v jenkins:/var/jenkins\_home \

-v /etc/localtime:/etc/localtime \

--name jenkins

jenkinsci/blueocean

如图为各参数含义解释

表3-1 Docker安装Jenkins命令参数解释

|  |  |
| --- | --- |
| -d | 后台运行镜像 |
| -p 8080:8080 | 镜像的8080端口映射服务器的8080端口 |
| -p 50000:50000 | 镜像的50000端口映射服务器的50000端口 |
| -v jenkins-data:/var/jenkins\_home | Jenkins工作目录为docker容器中的/var/Jenkins\_home |
| -v /etc/localtime:/etc/localtime | 同步容器与服务器的时间设置 |
| --name jenkins | 容器命名 |
| jenkinsci/blueocean |  |

（2）在浏览器输入<http://localhost:8080>进入Jenkins登录页面

（3）docker exec -it jenkins bash进入访问jenkins容器，访问/var/jenkins\_home/secrets/initialAdminPassword获取密码

（4）登陆后安装插件完成Jenkins环境搭建

## 本章小结

本章主要研究了持续集成系统的设计组成。首先研究和分析了持续集成系统的组成结构，然后描述了持续集成的具体流程，最后选区Jenkins作为持续集成服务器，并使用docker搭建Jenkins持续集成服务器。

# 基于持续集成在VR领域的实践

## VR生产环境的自动化配置脚本设计

在VR设备生产过程中需要对它的生产校准数据进行保存，涉及到了持续集成中环境配置的问题，这里采用脚本的方式实现自动化环境配置部署。

### 设计需求与方案

公司E3系列VR设备在生产期间需要通过一系列生产工具，对VR设备进行烧录、校准、功能测试、SN号写入等等操作，而这些操作的结果数据需要存储在MySQL数据库中。针对生产环境的各种要求做出如下应对：

（1）环境配置方式简单。设备生产代工厂的工人并不具有任何计算机的相关知识，他们只知道通过相关的说明按照步骤使用生产工具进行生产，而即使有公司人员去进行环境部署，也需要对多台设备进行操作，部署后还要进行环境的稳定测试，工作量巨大，且不易维护。通过脚本将配置步骤有序地实现，只需要通过简单的说明，工厂相关人员即可进行快速地环境配置，且脚本测试只需要进行一次，极大地减少了环境配置的工作量，也保证了环境运行的稳定性。

（2）脚本依赖少。工厂的基础计算机环境都是进行（Win7）初始化过的，没有安装任何编译软件或者相关执行组件，这就需要环境配置脚本必须保证能够在原生的环境中运行。通过分析，在工厂的环境中可执行的脚本是批处理脚本，后缀为.bat，使用batch语言编写。

（3）对各种可能失败的结果进行提示。脚本运行过程必然会因为各种各样的原因出现错误，如旧的数据库进程对安装数据的影响等等，这需要通过大量测试发现相关的问题，并在脚本中做出判断，并作出相关的提示供脚本使用人员进行操作。

（4）易于维护。生产环节的改变以及其他要求的改变会导致数据库环境做出改变，需要在脚本中进行相关修改，这要求脚本中的配置是易于寻找方便修改的。

总结来说，数据库环境的搭建脚本实现方案为：通过batch语言编写自动化脚本，脚本要求最好是一键式的，依赖少，易于维护，并有良好的操作提示。

### 设计过程

数据库安装使用的是MySQL的解压版，再使用脚本实现MySQL的安装，脚本分为安装卸载两个脚本。通过分析脚本主要分为如下模块：

1. 初始化。包含目录、MySQL初始化。
2. 配置参数。该部分参数即在维护中可修改的参数。
3. MySQL配置写入。
4. 依赖安装
5. 数据库安装
6. 环境变量配置
7. 卸载环境

详细步骤和实现方法如下所述：

1. 初始化和卸载环境

初始化是为了保证脚本运行顺利，首先要读取脚本所在目录，以在之后整个脚本中都是基于该目录的。Batch语言中使用%~dp0即可读取脚本所在目录。

在脚本运行前必须保证数据库的安装解压文件是纯净的，本质是将MySQL进行卸载，所以这里将初始化和卸载放在一起。在安装脚本中通过call命令来调用卸载脚本即可实现初始化。

卸载步骤如下：

1. net stop mysql停止MySQL服务
2. sc delete mysql卸载MySQL主要程序
3. taskkill杀掉mysqld.exe进程
4. del删除MySQL的配置文件my.ini
5. rd /s /d data删除MySQL的数据文件

2. 配置参数

为了方便修改脚本参数，将脚本中的常用参数通过set命令来设置，主要参数有：

1. 数据库服务器端口，一般为3306.
2. 数据库密码。
3. 数据库名。
4. 数据库表明
5. 全局变量主表路径。

3. MySQL配置写入

MySQL安装配置主要是通过my.ini文件设置的，通过如echo [mysql] >> my.ini可以将[mysql]写入到文本中，且不会覆盖之前的文本，而使用>是会覆盖的。需要写入的配置主要有：

表4-1 MySQL安装配置

|  |  |  |
| --- | --- | --- |
| 参数 | 解释 | 值 |
| default-character-set | MySQL客户端默认字符集 | utf8 |
| port | MySQL服务运行端口号 | 3306 |
| basedir | MySQL程序存放路径，常用来存放MySQL启动、配置文件、日志等 |  |
| datadir | MySQL数据库存放文件路径 |  |
| max\_connection | MySQL允许的最大连接进程数 | 200 |
| character-server-set | 数据库默认字符集 | utf8 |
| default-storage-engine | MySQL存储引擎 | INNODB |

4. 依赖安装

MySQL程序运行需要依赖Microsoft Visual C++ 2013 Redistributable(x64)，这是一个微软的运行库，该库支持命令行安装，下载vcredist\_x64.exe后，使用命令vcredist\_x64.exe /h可以查看命令行安装方式，如下图

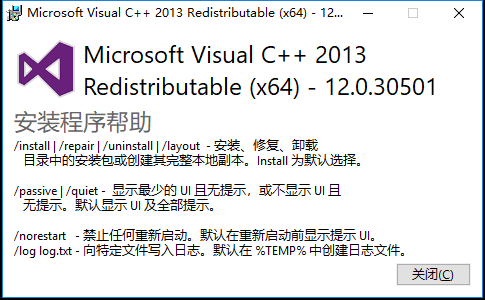


图4-1 vcredist\_x64.exe命令行安装

根据提示，为了快速安装该库采用静默安装，即使用vcredist\_x64.exe /passive命令。

5. 数据库安装

在将数据库配置文件和依赖库安装好后，就可以安装数据库了，使用命令mysqld –initialize和mysqld install即可安装实现数据库安装。但是安装好后的数据库是需要密码登录，为了能够快速进行数据库的各种操作，需要将数据库设置成无密码登录，主要操作如下：

（1）停止数据库服务。使用net stop mysql 2> nul停止数据库服务，由于net stop会在命令行窗口打印输出一些没必要的信息，这里使用2> nul隐藏输出。

（2）修改配置文件。将skp-grant-tables写入到my.ini文件中可以实现无密码登录。

（3）杀掉MySQL命令行工具。使用taskkill命令可以强制杀掉mysqld.exe。

数据库安装成功后，需要判断MySQL是否安装成功，判断依据是MySQL服务是否能够重新启动。由于判断依据不是一条特定的语句，而是服务启动结果，不能使用ifelse语句去进行判断，需要采用管道命令实现，主要使用的管道命令符是&&和||。&&表示前面语句执行成功运行后面语句，||相反为执行失败运行后面的语句。判断语句为net start mysql && goto USAGE || goto FAIL，即当MySQL服务成功启动则跳转到USAGE语句模块，启动失败跳转到FAIL语句模块，各模块描述如下：

（1）:USAGE。该模块为MySQL服务成功启动即成功安装后的操作。由于未进行数据库环境变量设置，数据库安装成功后必须进入到MySQL文件目录中的bin文件夹中才能够执行相关SQL语句和MySQL命令。在脚本中使用如mysql -e “use mysql”语句执行SQL语句，在此时就可以执行SQL语句了。由于之前是采用免密登录MySQL的，首先要设置root用户的密码，需要执行命令

mysql -e "use mysql"

mysql -e "update mysql.user set authentication\_string=password(%password%) where user='root';"

在更新root密码之后，需要使用flush命令刷新MySQL的系统权限相关表，否则会现拒绝访问，即执行

mysql -e "flush privileges;

然而，由于此时root用户已经有权限限制，后续建表将无法继续进行，相较之前而言，root用户的密码是可知的，只需要将root密码写入到my.ini配置文件中，最后通过执行命令

mysql --connect-expired-password -e "SET PASSWORD = PASSWORD('%password%');"

通过MySQL安全策略登录进去，之后即可执行SQL语句创建表和字段了。

到目前为止，关于MySQL的权限设置和表的相关创建都已完成，无需再对数据库进行其他读写操作，通过my.ini中写入root密码的操作并不是安全的。这里需要在配置文件中将密码设置删掉，batch语句无法指定删除文本中的某一行，只能通过重新写入配置对my.ini作出更改。

（2）:FAIL。该模块为MySQL服务启动失败后的操作。服务启动失败即MySQL未安装成功，此时需对可能引发失败的原因做出提示，提醒使用者重新尝试运行脚本，且为了保证安装程序仍然纯净，要使用call调用一次卸载脚本。

6. 环境变量配置

通过前面的步骤执行下去，MySQL数据库已经基本安装完成，但是只能在MySQL的bin文件夹中运行，所以需要将bin添加到Windows的环境变量中，即可在全局环境中运行MySQL数据库服务。

一般手动添加环境变量只需要修改Windows中的path即可，然而脚本中并非如此。使用脚本添加path需要三步：

1. 获取原来的path；
2. 将要填加的path和原来的path通过分号合并在一起；
3. 将合并后的path赋值给环境变量中的path；

总结来说主要需要获取原来的path，然后再替换path。在batch脚本中可以直接通过%Path%来获取path，再用过setx命令重新设置path。但是通过实践发现，通过这种方式获取到的path是包含了用户和系统两个path的，而setx命令只对用户变量做出修改。在这种情况下，修改环境变量的结果是将用户path、系统变量path和添加的的path都作为用户变量的path，修改后的用户变量path将变得十分臃肿，严重破坏了之前的环境变量的值。且path是有长度限制的，如果path足够长，那么限制长度后新添加的path将不会被写入。

通过研究发现，Windows中的系统和应用程序的设置信息都存储在注册表当中，注册表其实是一种windows特有的数据库，类似传统的数据库，注册表中有字段名、类型和相应的值，而path为字段名，修改path需要修改改path对应的值。相应的，系统环境变量位于HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\Environment\，用户环境变量位于HKEY\_CURRENT\_USER\Environment，这里只修改用户环境变量，如图为Windows注册表编辑器查看情况

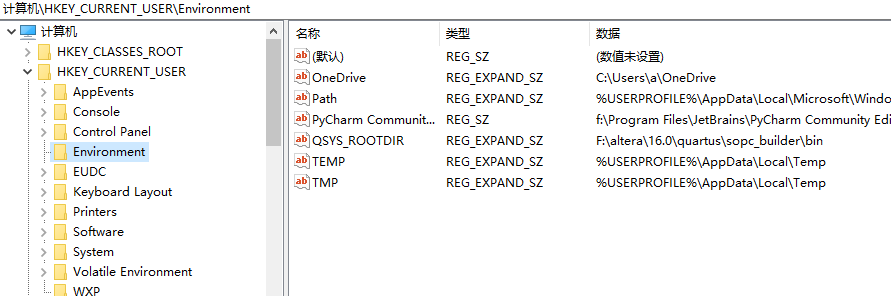


图4-2 Windows注册表编辑器

在batch脚本中，可以使用reg query命令查询注册表的值，查询条件为路径和字段名，即用户环境变量的查询条件为HKEY\_CURRENT\_USER\Environment和Path。查询出来后的输出为字段名+空格+类型+空格+值，但只需要获取path的值，需要对查询输出进行筛选。在batch语句中使用for循环可以按照特定符号分割文本，所以可以根据空格将输出分割为字段名、类型、字段值，最后再获取字段值，具体代码如下

for /f "tokens=1,2,\* delims= " %%a in ('reg query "%regpath\_%" /v %evname\_%') do (

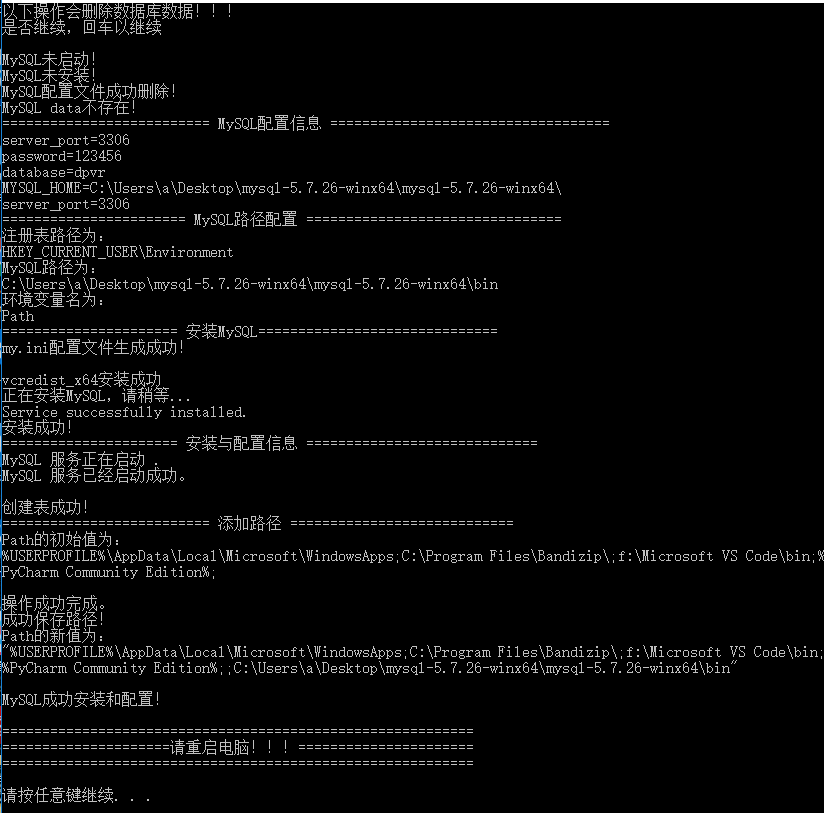
set pathall\_=%%c

)

delims表示分隔符为空格，tokens表示将内容分割为三部分，且将三部分分别赋值给%%a，%%b，%%c，最后再将字段值部分赋值给变量pathall\_。在添加path时需要首先确认该系统中是否已经存在path，如果存在则不添加并作出相应提示；如果不存在则使用reg add 命令修改path即可。至此，关于MySQL的一键配置环境完成。

### 结果分析

经过测试，脚本适用于5.7版本的MySQL数据包安装，低版本会出现无法创建数据文件的错误，从而导致无法成功安装。在安装时，必须保证系统环境中不存在已经安装MySQL数据库环境，否则会因为端口冲突造成数据库无法成功安装。通过注册表添加环境变量多数情况下环境变量不会立即生效，需要重启电脑，脚本成功运行结束后会做出提示。如图为MySQL安装成功的脚本运行情况。



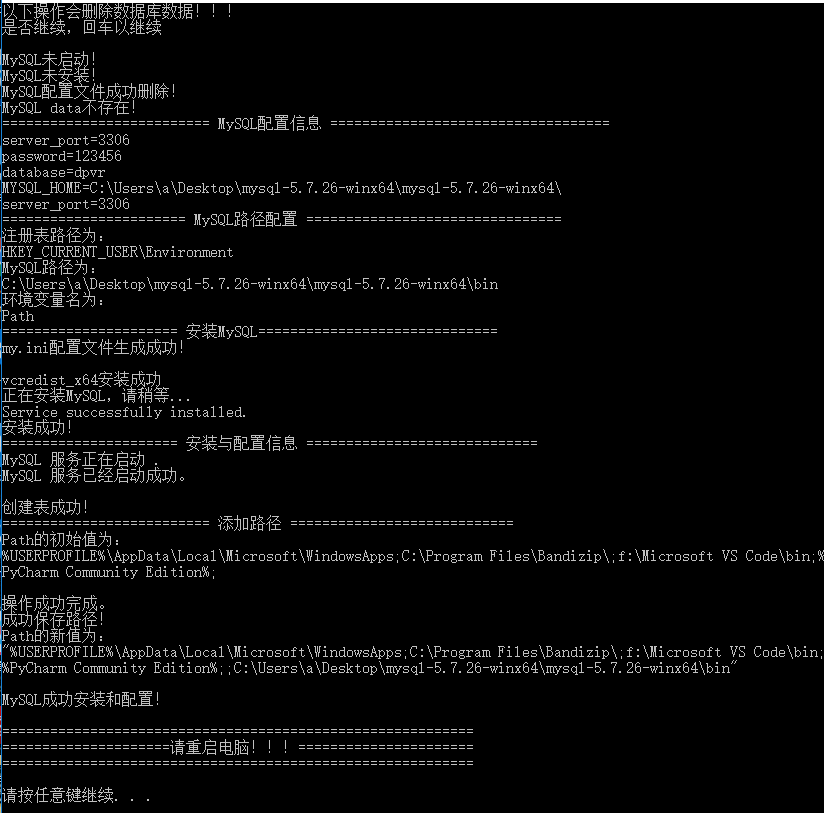


图 4-3 脚本运行成功情况

数据库安装成功后环境变量可以在Windows设置中查看，如图

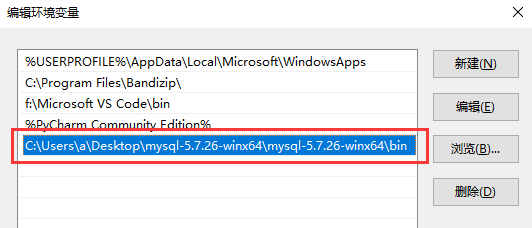
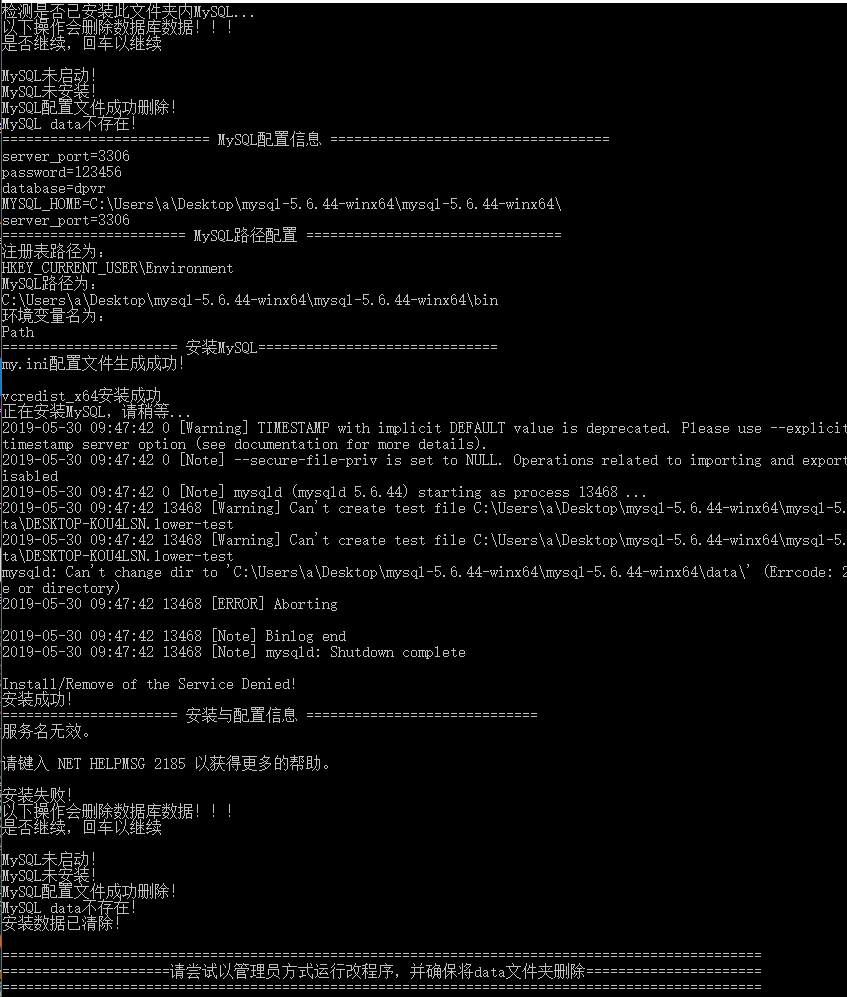


图4-4 环境变量查看

在MySQL服务无法成功启动时，MySQL会提示输入命令查询帮助，之后是脚本做出的相关失败提醒。根据提示可以看出，在安装过程中MySQL服务无法启动，data文件未成功创建，配置文件成功创建，在失败后之前所有创建的文件最后都被清理。如图为脚本运行后安装失败的情况。



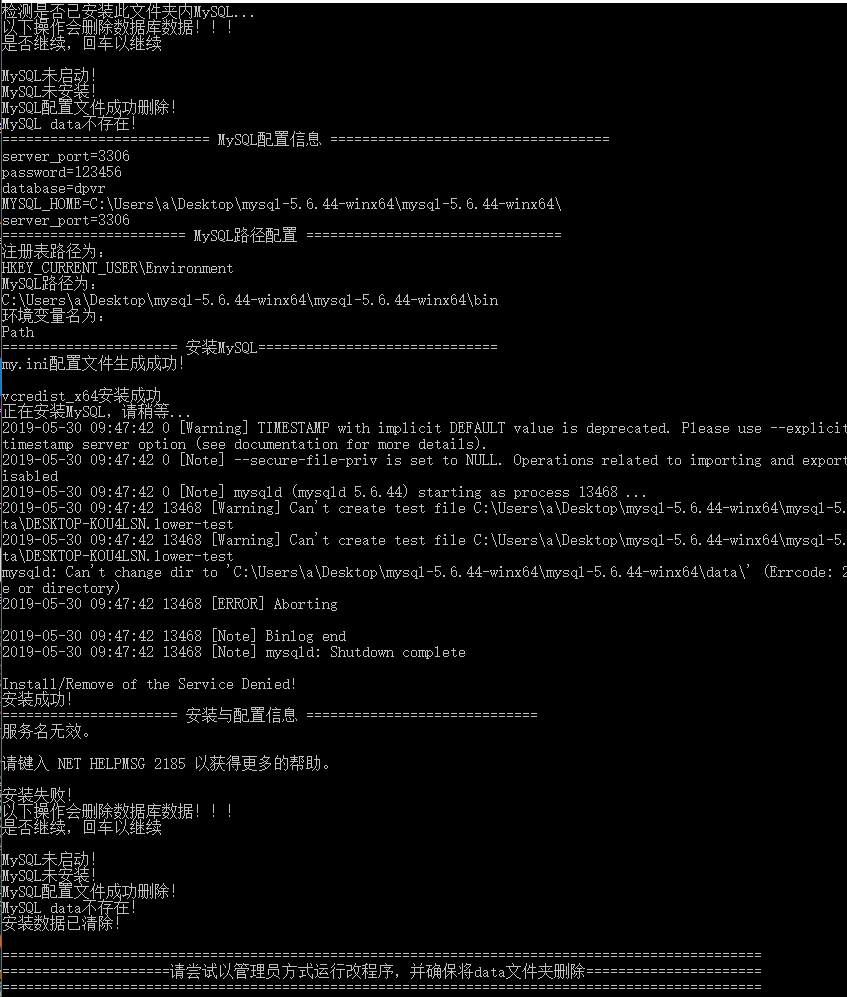


图4-5 脚本运行失败情况

如图，执行命令mysql -uroot -p输入密码后即可登录mysql的root用户。

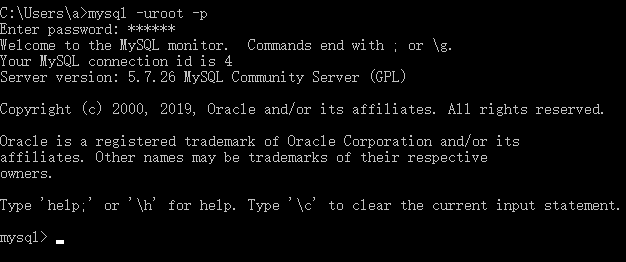


图4-6 登录MySQL

进入MySQL环境后，即可通过SQL命令查询数据库内容，如图可以看到脚本已经自动创建了dpvr的数据库。

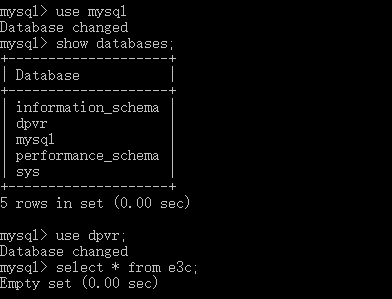
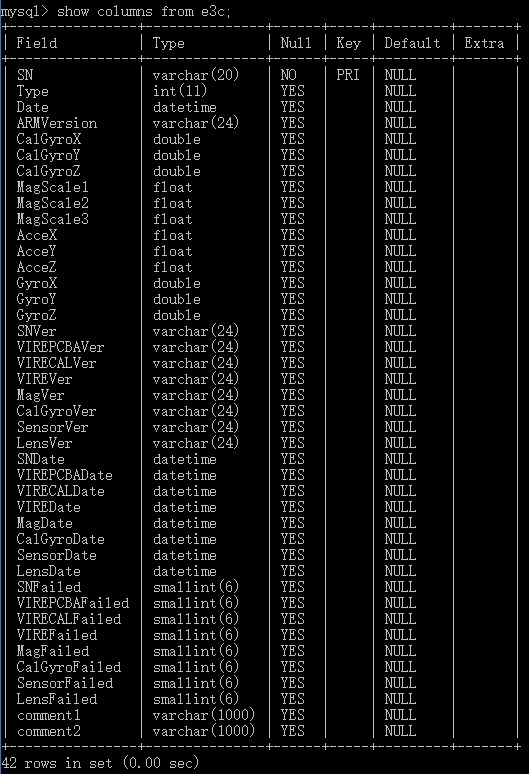


图4-7 数据库查询

如图，脚本自动创建的数据库表文件为e3c，由于目前并未录入任何数据，数据表中数据为空，可以通过show columns from e3c查询表的字段名，如图为脚本自动创建的VR生产数据表的各个字段及类型。



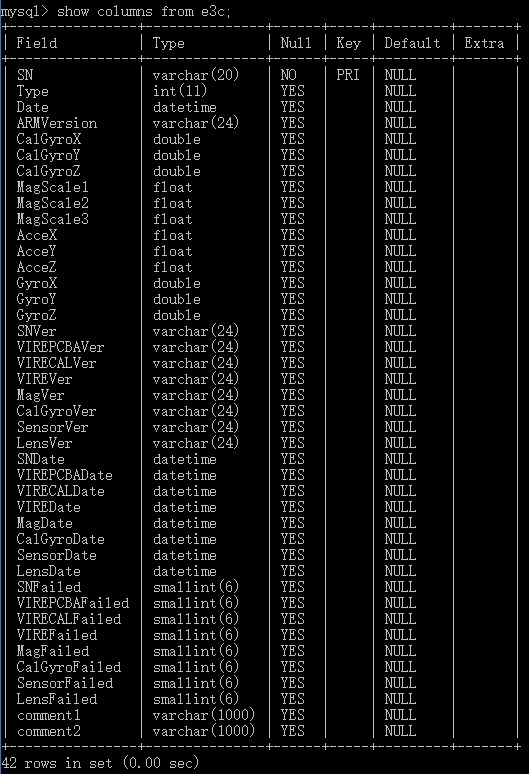


图4-8 数据表字段和类型

### 总结

作为持续集成中的一环，环境配置决定了持续集成的基础运行能否稳定。在VR设备生产过程中，其实就是一个集成的流水线，只是不同于软件开发中的持续集成，生产过程的迭代并无工作量的增加，所以尽可能地迭代更快地生产VR设备才能使收益最大化。zaiVR设备生产过程中，刷机为构建过程，功能测试为测试过程，这些都依赖于生产工人的生产能力、和生产工具的稳定性。保障生产工具的稳定性即保证生产环境的稳定性，通过自动化的环境配置，实现了环境配置的最快部署，加快了生产流程，同时自动化环境配置可以防止配置的遗漏，保证整个生产环境的持续稳定。

在编写自动化配置脚本时，通过深入了解批处理，批处理其实可以应用于很多日常的操作当中。批处理脚本只依赖Windows的原生环境，且仅仅几条简单的命令即可解放大量的手动操作。很多的开发工具都预留了命令行操作，可以通过在批处理脚本中写入命令对开发工具进行快速操作。

在Windows中，注册表中保存了Windows所有的软件配置项，部分软件可能未开放的设置都可以通过修改注册表强制进行修改。对注册表的修改更适用于在脚本中使用，适合自动化，但需要对表中变量值足够了解，否则修改注册表是非常危险的操作。

在作为开发人员编写脚本时，必须对脚本运行的可能性做出各种判断，针对脚本的使用者做出相应的设计，傻瓜式的设计是最能够适用所有人的。无论是普通用户，还是生产工人或者是其他开发人员，快速地自动化流程和良好的提示都能使得脚本的体验得到提升，用户可以通过提示向开发人员寻求解决方案，开发人员也可以根据提示快速调试解决问题，也可以对脚本进行二次开发。当然，任何自动化都是固定的，但是生产流程往往是多变的，自动化脚本必须能够快速适应需求进行更改，在编写过程中发现，将平凡使用、需要修改的脚本命令或参数放在一起，并作出良好的注释是最佳的方式。

## VR应用测试情况自动报表脚本设计

在测试VR设备过程中，开发人员和测试人员是通过项目管理软件进行bug提交与解决沟通的，公司需要一个自动报表系统来更好地对bug数据进行统计。

### 设计需求与方案

本人实习所在公司使用“禅道”项目管理工具对开发的项目进行设计、创建、开发、测试、维护等等，如图为项目各参与人员的工作流程图：



图4-9 禅道项目管理流程

在项目管理过程中，测试人员将bug提交到禅道上，开发人员通过查看bug具体描述对bug进行修复，在禅道上对bug进行状态更新。如图为bug各状态信息。



图4-10 Bug各状态信息

虽然禅道上对bug的信息展示非常具体，但是对于大多数项目参与人员来说，当bug数量越来越多时，关注的重点往往是严重程度最高比如一级和二级bug，大量的三级bug累积下来越积越多。而项目经理在对项目进行可行性评估时，首先是以一级和二级bug的数量来评定版本是否可以发布，其次是三级bug的数量。因而往往存在一个问题，较严重的bug被解决后，在项目经理进行版本代码复核时，往往会因为三级bug过多而将版本打回。所以，除了对严重bug提高关注，还需要bug的整体结构分析。通过每天的代码记录，将各类登记的bug统计下来，并通过图表对bug数量的变化进行可视化的展示，并通过邮件将代码统计情况每天发送给所有的项目参与人员。

开发人员可以根据每天收到的报表查看当前bug的数量与发展趋势，除了着重对较严重bug进行关注外，还可以了解目前三级bug数量的变化情况。这样，开发人员可以通过bug的数量与测试人员进行沟通，了解bug数量变化的原因，从而对出现问题的项目模块快速核查，可以一次性解决大量的低级bug。开发人员也可以通过图表所展示的bug信息，进而分析bug数量变化对自己未来开发工作重心的影响，提前对自己的工作内容进行调整，提高开发的效率。

项目经理可以根据报表中bug的数量统计，大体了解目前项目的发展情况。当bug数量大量产生时，说明目前项目整体仍然处于开发初期时的不稳定状态，那么此时项目经理可以根据bug情况与测试和开发人员进行沟通，了解目前产生大量bug的具体模块和问题所在，项目经理可以根据这些情况对项目参与人员的工作内容进行调整，缩短项目的不稳定期，快速提高项目的开发进程。当新增bug中严重bug数量较少，三级bug数量较多时，说明此时项目已经较为稳定，仍然存在一些影响体验的小问题存在，当然这些小问题无法被忽视，用户的体验往往都是集中在细节的体验上的。这种情况下，项目经理可以开始考虑关注大量的三级bug，通过分析三级bug产生的模块，与产品经理进行沟通，讨论模块设计的可行性，要求产品经理在下一个版本中做出相应的修改和设计。当bug数量持续减少时，说明此时bug的解决速度远远大于bug的产生速度，此时bug已经基本不会产生，项目整体上已经稳定，项目经理可以将关注度放在开发人员身上，开发人员进行沟通，讨论遗留bug的影响情况，对遗留bug的解决优先级进行调整，进而在bug逐渐减少的一定界限内，对项目版本的发布时间进行提前安排。

测试经理可以根据报表中的bug数量统计，了解目前测试人员的测试情况。在项目版本刚刚提测后，测试经理可以根据bug初期的产生情况决定该版本是否可以继续进行测试，如果bug产数量变化十分大，那么测试经理可以与开发人员和项目经理进行沟通，停止测试并打回改版本，要求开发人员对项目版本修改模块进行重新开发，减少测试工作量。在项目进行到后期，bug数量趋于稳定时，测试经理可以对项目测试参与人员的工作进行重新安排，将一部分测试人员安排到其他项目组去，一部分测试人员继续进行项目复核。

整体来说，bug数量的多少也可以侧面体现出项目参与人员的工作情况，可以用于对项目参与人员的工作评定中。项目参与人员也可以通过bug数量的情况相互激励，从精神上提升他们的工作体验。

所以需要一个自动bug统计报表系统，该系统主要具备如下功能：

（1）抓取禅道bug数据。项目的bug情况都记录在禅道上，但是禅道并没有相关的bug趋势相关的统计功能。禅道系统是基于网页的，一般来说，抓取网页的数据最好的方法是使用爬虫，可以通过脚本编写爬虫抓取bug情况信息，并存储下来。

（2）数据存储。将bug数据抓取下来后，需要每天抓取到的数据存储在数据库中，一般来说选取MySQL作为数据库来读写数据是最方便的，但是生产的数据表需要能够在邮件中方便查看，那么需要一种邮件系统支持在线显示的数据文件，Excel作为数据文件符合这种要求。

（3）绘制数据图表。将每天的数据存储到Excel表中后，需要将数据再次读取出来，按照时间进行绘制曲线图、柱状图等等，并且绘制出的图表可以保存为图片。

（4）自动发送邮件。生成的数据文件和图表图片以及相关统计情况需要通过邮件发送到项目参与人员中，最后可以通过Jenkins定时任务将每天定时发送。

### 设计过程

通过对自动报表系统的要求和设计的分析，本项目采用Python脚本实现自动抓取bug数据、读写数据、绘制统计表、发送邮件的功能，再通过Jenkins的定时任务功能每天运行一次脚本，实现定时发送的功能。

相应的，本项目的python脚本按照功能分为如下几个python文件：

表4-2 报表脚本文件

|  |  |
| --- | --- |
| 脚本 | 功能 |
| data2file.py | 数据处理 |
| send\_email.py | 邮件发送 |
| settings.py | 脚本相关设置 |
| zentao\_driver.py | 禅道抓取数据 |
| report\_main.py | 主函数 |

本脚本将除了report\_main.py主函数脚本外的其他脚本autoreport文件夹中，作为一个autoreport工具包，其他开发人员可以自己编写主函数调用工具包中的脚本中的类和方法。Python是一门面向对象的脚本语言，工具包中的脚本都创建了类和方法，各脚本中创建的类如下表：

表4-3 报表脚本创建类

|  |  |  |
| --- | --- | --- |
| 脚本 | 类 | 功能 |
| data2file.py | Data2file | 数据处理 |
| send\_email.py | SendEmail | 邮件发送 |
| settings.py | Settings | 脚本相关设置 |
| zentao\_driver.py | XPath | XPath映射 |
| ZenTao | 禅道抓取数据 |

如下按照脚本模块介绍脚本的设计过程。

1. settings.py设置相关

settings.py脚本中创建了Settings类，该类中只创建了类属性，即脚本设置的相关参数，这些参数是脚本使用者可以修改的设置，使用者可以根据自己的情况对各参数进行修改。所有脚本都是基于settings.py脚本中的参数驱动的，必须保证个脚本都对该脚本进行了调用声明。下表为该脚本的Settings类属性介绍：

表4-4 Settings类属性

|  |  |  |
| --- | --- | --- |
| 参数 | 描述 | 备注 |
| time\_now | 目前时间 | strftime('%Y/%m/%d') |
| mail\_host | SMTP服务器 | smtp.mxhichina.com |
| mail\_user | 邮箱地址 |  |
| mail\_pass | 邮箱密码 |  |
| sender | 发送者 | name<usr@mail.com> |
| receivers | 接受者 | name<usr@mail.com> |
| cc | 抄送者 | name<usr@mail.com> |
| subject | 邮件标题 |  |
| sleep\_time | 模拟点击间隔 |  |
| in\_address | 内部禅道地址 |  |
| in\_account | 内部禅道账户 |  |
| in\_password | 内部禅道密码 |  |
| ex\_address | 外部禅道地址 |  |
| ex\_account | 外部禅道账户 |  |
| ex\_password | 外部禅道密码 |  |
| folder | 数据文件储存位置 |  |
| xls | 数据表文件名 |  |
| num\_table\_data | 邮件显示数据条数 |  |
| self.num\_img\_data | 折线图统计数据条数 |  |
| tinify\_key | 图片压缩api的key |  |
| bug\_type | Bug类型 |  |

2．zentao\_driver.py禅道数据抓取相关

zentao\_driver.py中创建了两个类：XPath和ZenTao，这两个类都是基于selenium的 webdriver创建的。

Selenium是以一套web自动化测试框架，在python中拥有selenium相关的脚本库可以调用，除此之外还需要下载chromedriver.exe作为驱动。如图为selenium支持的浏览器，本脚本采用了chrome浏览器。

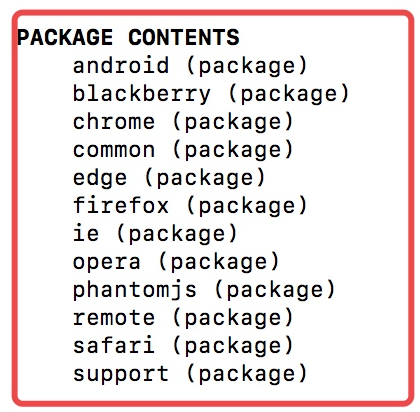


图4-11 selenium支持浏览器

如图为chromedrvier的部分镜像，chromedriver必须与chrome浏览器的版本相对应，否则将无法启用。

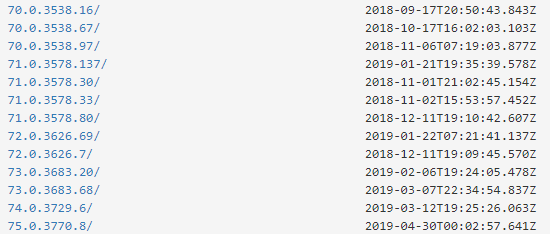


图4-12 webdriver对应chrome版本

Selenium除了用于web自动化测试外，常常用作网络爬虫自动抓取数据。Selenium抓取数据主要的方式是通过定位元素，再对元素进行模拟点击、数据获取等操作。Selenium定位元素有很多方式，如通过元素的id、text以及xpath。这些定位方式并不是一成不变的，需要根据网页的具体情况进行分析，一般来说需要结合起来一起使用进行元素的精准定位。一般来说，网页元素的路径是不变的，那么使用xpath进行元素定位是最为迅速地。Xpath通过特定的语法对元素的路径进行描述，如此看来我们需要了解xpath的语法，看起来是很复杂的。但是chrome浏览器可以自动生成元素的xpath，相对于通过text或id定位元素。xpath是唯一的，不必考虑是否会定位到其他元素去。

部分网站为了反爬虫或者出于其他安全因素会对元素的路径进行实时更新，这种情况下xpath在每次页面更新都会改变，定位无法生效。此时，就需要通过id和text以及其他分析到的变化条件对元素进行综合性精准定位，这种方式是非常复杂的。当然，当页面元素较少时，元素的id一般是唯一的，直接使用id定位即可。

通过测试，禅道的元素路径是不变吧，所以本脚本中大量使用了xpath进行定位。但是，在使用xpath编写脚本后，如图由于xpath中存在大量特殊字符，在脚本中显示十分混乱，对脚本的可读性造成很大影响。

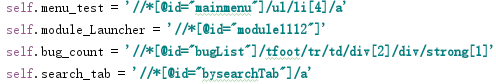


图4-13 脚本中的xpath

如此，脚本中对xpath创建一个单独的类，通过类属性对xpath进行映射，且XPath类的属性变量名通过一定的格式编写，开发人员和使用者只需直接使用映射后的XPath类属性，无需考虑xpath语法。脚本要求其他开发人员在进行脚本修改时在每次使用其他 xpath时将其添加到XPath类中按照格式进行映射。

在zentao\_driver.py脚本中还创建了一个类ZenTao，该类主要创建了几个抓取数据的方法，这些方法抓取禅道不通过项目、模块中的bug数据。下表为ZenTao类的属性和方法的相关描述：

表4-5 Zentao类属性

|  |  |  |
| --- | --- | --- |
| 属性 | 描述 | 备注 |
| settings | 设置 | 使用Settings类实例 |
| driver | webdriver | 使用Chrome |
| xpath | XPath实例 |  |
| sleep\_time | 模拟点击间隔 | 使用Settings类属性 |

表4-6 ZenTao类方法

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 参数 |
| click\_by\_xpath | 通过xpath定位并点击 | xpath |
| click\_by\_id | 通过id定位并点击 | \_id |
| click\_by\_text | 通过text定位并点击 | text |
| bug\_count | 获取bug数量 |  |
| read\_table | 获取bug详细信息 |  |
| login | 登录外部或内部禅道 | addr, account, password |
| bug\_search\_case1 | 获取内部禅道bug数据 | project, module\_xpath |
| bug\_search\_case2 | 获取一级bug信息 | project, module\_xpath |
| bug\_search\_case3 | 获取外部禅道bug数据 | project |

如表所述，ZenTao类方法分为四类：

（1）click\_by\_xpath、click\_by\_id和click\_by\_text通过定位元素进行点击，这些方法出了点击动作外，还加了延时函数，参数为settings中的sleep\_time，以达到模拟点击直接存在间隔。sleep\_time的时间需要经过特定的测试以达到最好，如果太小，相邻两次的点击间隔太短，造成下一次的元素无法定位，脚本报错；如果太大，响铃两次的点击间隔太久，造成脚本运行时间过长，占用系统资源。通过测试sleep\_time的值设为1s较为恰当。

（2）login方法通过定位元素并使用sendkey发送字符串，可以实现自动登录。

（3）bug\_count和read\_table都是对bug的信息进行统计的方法。bug\_count的获取方式比较简单，只需要定位元素后使用get\_attribute('innerHTML')获取元素内容，获取后将字符串取整用作之后对数据处理。经过测试发现，当bug数量为0时，不会显示bug数量，bug数量无法获取脚本运行错误。这里使用了try…except对运行结果进行判断，当无法和获取bug数量时，打印报错信息，并将bug数量置零。

read\_table是获取bug的详细情况，即获取禅道上的bug表信息，获取到的数据为一个二维数组。通过使用chrome浏览器审查元素，bug表的id为“buglist”，class为“text-center”，通过循环，按照tag为“td”读取每行和每列的数据，最后将读取到的数据存储在一个二维数组中。

（4）bug\_search\_case系列方法是模拟用户点击的流程，通过调用前面编写的方法最终获取到需要的bug数据。该系列方法要求使用者自行调用前面的方法进行编写，当然为了应对目前公司的需求，本脚本创建了三个bug\_search\_case方法。大多数的bug数据获取步骤大同小异，一般步骤为登录，选择的项目，按照需求检索，获取bug数据。根据需求，多数数据的获取步骤只是项目与检索模块不同，所以为了方便函数复用，bug\_search\_case方法编写建议预留项目和模块的参数。

3. datafile.py数据处理相关

data2file.py中创建了Data2file类进行数据转文件的相关处理。数据处理主要有两类：Excel数据表的读写和绘制数据图表。Data2file的属性和方法如下表：

表4-7 Data2file类属性

|  |  |  |
| --- | --- | --- |
| 属性 | 描述 | 备注 |
| settings | 设置 | 使用Settings类实例 |
| folder | 文件保存路径 | 使用Settings类属性 |
| report\_name | Excel保存路径 | 使用Settings类属性 |
| row\_type | 行标（时间） | 使用Settings类属性 |
| col\_type | 列标（bug类型） | 使用Settings类属性 |
| tinify.key | Tinify API key | 使用Settings类属性 |

表4-8 Data2file类方法

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 参数 |
| bug\_list2xls | 将bug情况表写入Excel | table, xls, sheet |
| data2xls | 将bug数量写入Excel | data, sheet\_index |
| Read\_xls | 读取Excel中的bug数据 | sheet\_index |
| data2img | 绘制bug曲线图 | title, rows, sheet\_index |
| data2img\_bar | 绘制bug指派柱状图 | title, x, y |

当获取数据后，首先要将数据写入到Excel中。根据需求，bug数据主要有两类：每日bug统计量和每日bug详情。其中bug统计量需要每天在原有Excel的基础上进行添加，再对所有数据进行统计分析；bug详情只需要关注每天的bug指派、优先级等情况，需将之前的数据覆盖掉。

在python中使用xlrd、xlwt、xlutils函数库可以实现对Excel数据表的读写。Xlrd主要用来读取Excel中的数据，但是不能修改其中的数据，而xlwt用来将数据保存到Excel中，但是数据会覆盖，所以要使用到xlutils.copy创建副本对其修改并保存。

在修改获取Excel表中的数据后，使用pyecharts绘制数据统计图。pyecharts是基于 echarts.js的api编写的python工具包，相比较而言使用echarts.js需要手动编写js文件来设置图标样式，再通过编写python脚本调用api；Pyechaets的图标样式是固定的，该工具包已经集成了echarts.js的各类型的图标，非常方便调用。更方便的是，Pyecharts存在将生成的图表渲染为html和png的方法，只需要简单掉调用该方法即可。所以脚本中采用了pyecharts作为图表绘制库。

pyecharts渲染后的图片的需要插入到email正文中，通过测试发现，渲染后的图片每张达到5M左右，这导致每天bug统计邮件非常大，所以需要在渲染后对图表文件进行压缩。脚本中使用tinify工具包进行图片压缩。

Tinify工具包通过调用TinyPNG的api进行图片压缩。TinyPNG是一个可以在线压缩PNG格式图片的网站，经过测试，生成的图表文件经过压缩后只有几百K，压缩效果非常好。如图TinyPNG的官方文档中提供了api以供图片的自动上传压缩。

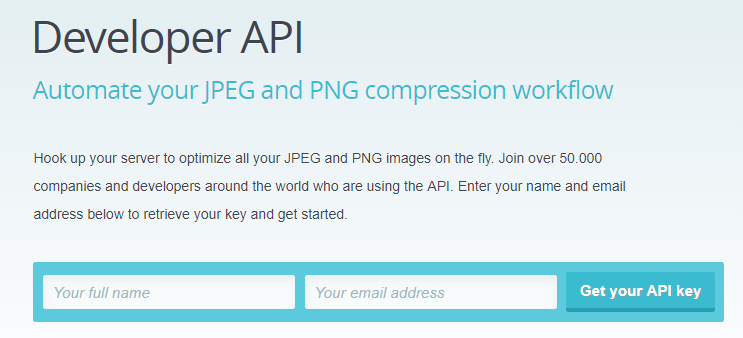


图4-13 TinyPNG API

用户可以通过姓名和邮箱获取一个API key，每个邮箱可以免费获得每个月500次的压缩次数，完全够用于自动压缩图表。在python中Tinify很好的封装了该api，只需要将获取的API key通过 tinify.key导入即可。tinify.from\_file(img\_path).to\_file(img\_path)方法可以将PNG图片直接压缩。

4. send\_email.py发送邮件相关

send\_email.py脚本中创建了SendEmail类实现将文字、图片、附件添加到邮件中并发送。SendEmail类的属性和方法如下表：

表4-9 SendEmail类属性

|  |  |  |
| --- | --- | --- |
| 属性 | 描述 | 备注 |
| settings | 设置 | 使用Settings类实例 |
| msgRoot | 邮件对象 |  |
| sender | 发送者 | 使用Settings类属性 |
| receivers | 接受者 | 使用Settings类属性 |
| msg\_html | 邮件正文 |  |
| msgAlternative | 邮件附件 |  |

表4-10 SendEmail类方法

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 参数 |
| read\_addr | 邮件地址格式化 | name\_email\_list |
| text2html | 文字转html | text, weght |
| table2html | 表格转html | data, width |
| img2html | 图片转html | img, tag, width, height |
| add\_html | 将html正文添加到邮件 |  |
| add\_attach | 添加附件 | file |
| send\_email | 发送邮件 |  |

使用python发送邮件主要使用了smtplib、email.mime、email.header和email.utils函数库，其主要思路为创建一个邮件对象msgRoot，该对象包含标题、发件人、收件人、正文和附件，该对象是一个字典类型，其相应的key和value如下表：

表4-11 msgRoot的key和value

|  |  |  |
| --- | --- | --- |
| key | value | 描述 |
| Subject | Header() | 标题 |
| From | Formataddr() | 发件人 |
| to | to\_addrs | 收件人 |
| cc | cc\_addrs | 抄送 |

邮件正文是以html格式编写的，其中包含文字、表格、图片等等。最后将正文和附件一起添加到msgRoot对象中，整个邮件内容就完成了。

邮件发送主要是使用邮箱代理服务器发送的，该脚本使用SMTP SSL验证，请求端口为465，使用SMTP服务器。登录SMTP服务器后，使用sendmail方法即可发送邮件，其参数为发件人、收件人和msgRoot。

当然，邮件发送常常会因为各种原因出现问题，开发者需要通过错误码进行分析出错原因，提出解决方案，smtplib库中对相应的错误码进行了规定，在脚本中使用try…except可以进行错误提示，具体报错对应如下：

表4-12 smtplib报错提示

|  |  |
| --- | --- |
| 错误 | 描述 |
| SMTPConnectError | 连接失败 |
| SMTPAuthenticationError | 认证错误 |
| SMTPSenderRefused | 发件人被拒绝 |
| SMTPRecipientsRefused | 收件人被拒绝 |
| SMTPDataError | 数据接收被拒绝 |
| SMTPException | 邮件发送失败 |

5. report\_main.py主函数

前面已经将脚本的各个模块都完成了，在主函数就要对所有模块进行组装，实现完整的脚本功能。在主脚本中，根据需要统计的项目和模块可以单独划分函数，以防止主函数过于冗长，不利于修改，主脚本中使用bug\_count\_report系列来表示实现需求的响应函数，这些函数都可以单独运行，相互直接不会有所影响，所以建议开发者在添加新的统计方法时将方法按照bug\_count\_report的格式进行编写，以实现脚本的可持续维护。

在主函数需要创建Settings、Data2file和SendEmail实例，并将这些实例传递到各个方法函数中。在每个方法函数中，只需创建ZenTao的实例，使用这些实例可以通过调用其方法快速实现bug数据抓取和数据统计和邮件发送。至此，脚本所有相关功能都已完成。

最后，只需要将脚本放在Jenkins服务器中，通过每日定时构建实现每日统计bug情况并自动发送邮件的功能。

6. 脚本移植

脚本编写测试成功后，需要放在Jenkins服务器中，但此时python缺少相应的函数库从而无法运行。需要在脚本编写环境中运行pip freeze >requirements.txt生成依赖文件，再到Jenkins服务器中运行pip install -r requirements.txt即可将所有依赖安装完成。如图为生成依赖列表。

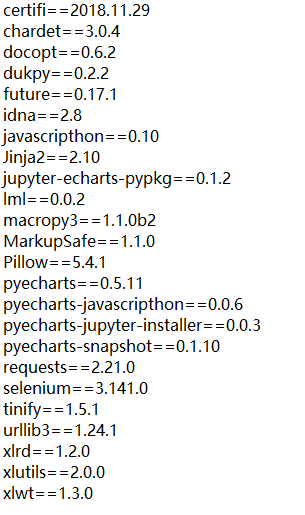


图4-14 脚本依赖

### 结果分析

如图为4月26日自动发送的邮件，邮件中标题、收件人、发件人、抄送均显示成功。



图4-15 邮件信息

该天P1Pro项目一级Bug指派情况如图，邮件中使用表格将一级bug的编号、严重程度、优先级、标题、bug状态和指派人列出，并使用柱状图对各指派人的一级bug进行统计。

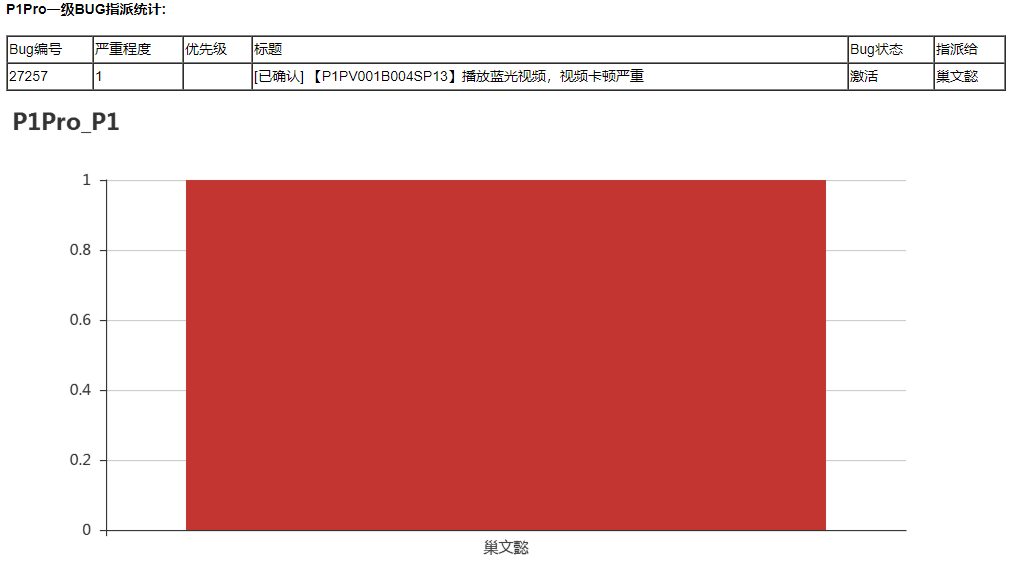


图4-16 一级Bug指派

如图为该天P1pro Launcher模块的Bug情况统计，邮件中的表格取近四天的统计数据，折线图取近30天的统计数据。从表中可以具体看出仅四天各各类型bug的具体数量变化，可以明显看出在4月25日到4月26日期间关闭了100多个bug，说明当天集中工作对遗留的已解决的bug进行了验证，并将其关闭。从折线图中可以看出近30天的bug数量变化情况，在4月12日到4月14日之间三级bug大量增加，此时应该是新版本刚刚发布，且改动较大，新问题较多。而在4月25到4月26日这些三级bug被大量解决和关闭。

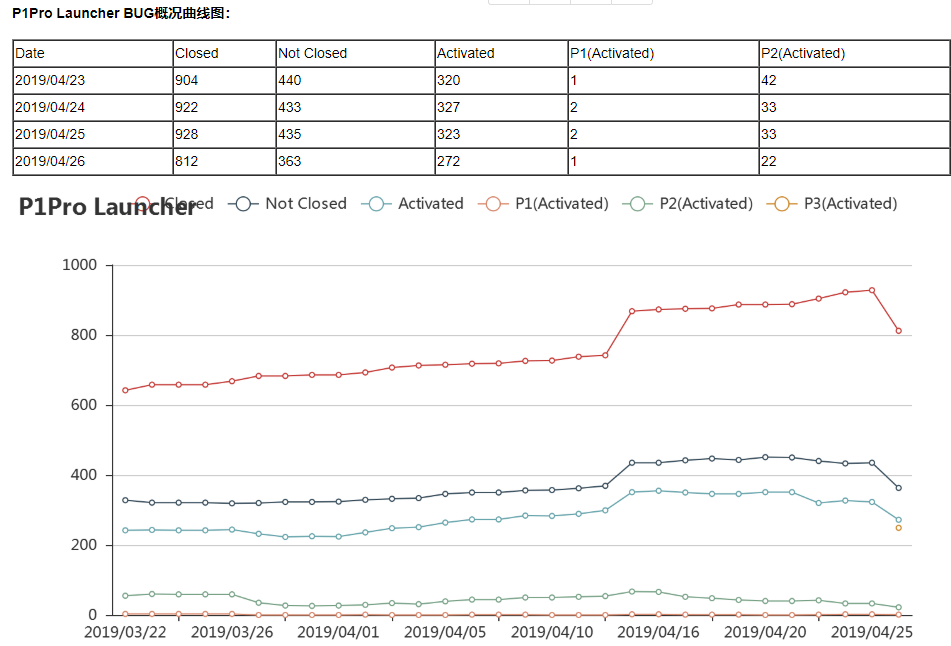


图4-18 P1Pro Launcher Bug统计

如图为P1Pro一体播控Bug概况图，可以看出该项目整体上来看近30天的Bug数量较为平稳，但是每天的Bug数量仍然在上下浮动，说明目前项目整体功能已经完善，但是仍然在持续开发，版本更迭的改动并不是很大。

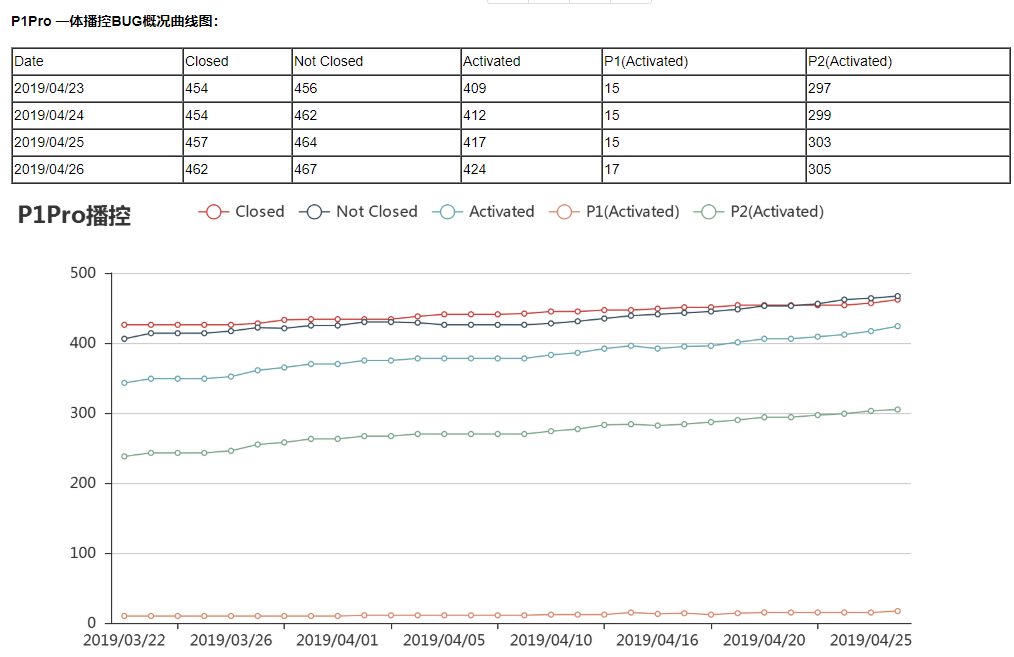


图4-19 P1pro一体播控Bug统计

如图为研发总监在查看邮件后发现二、三级bug数量回升，督促项目组开发人员留意bug的修复。

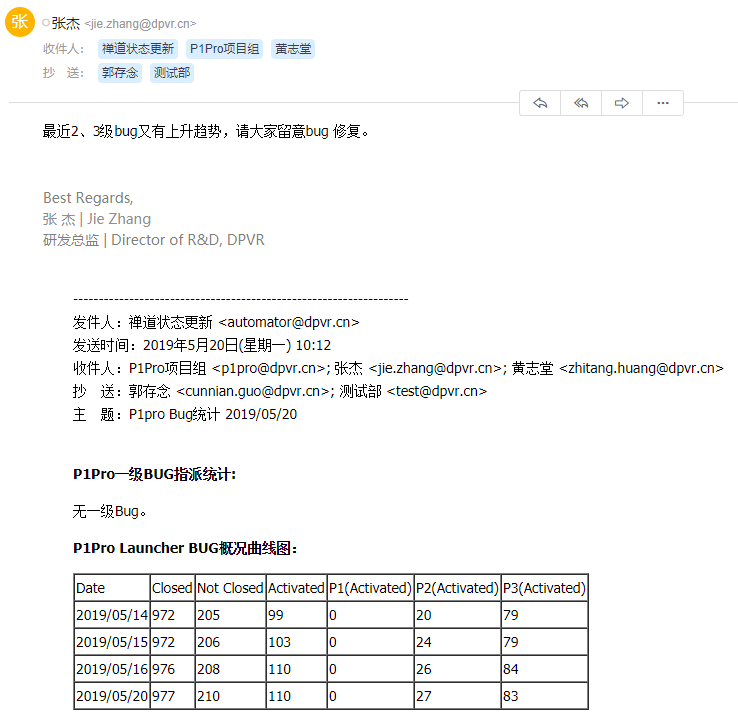


图4-20 研发总监邮件

### 总结

作为持续集成中的一环，通过邮件发送日志信息可以对持续集成后的相关情况进行及时通知，而通过邮件发送每日报告，可以对持续集成的多次迭代情况进行整体性分析。通过分析项目在进行多次迭代后的Bug数量统计情况，可以很好地分析出目前项目的进展，项目的各个参与人员都对项目的大致情况进行了解。

除了对该脚本作为持续集成一环的理解外，通过亲自编写脚本更能体会到持续集成的重要性。在编写脚本过程中，功能的实现其实数十天就可以完成，然而在需求逐渐增加时，不仅仅需要考虑的是功能的实现，更多考虑的是代码的可维护性。在编写代码初期，虽然需求的具体功能已经实现，但是所有代码都集中在单个脚本中，许多功能代码都要重复编写，参数在脚本中分布混乱，如XPath等包含特殊字符的字符串在脚本中随意防止，脚本中参数和功能也没相关的注释。当有新的需求时，简单的一个需求修改如模拟点击间隔的时间调整，需要在脚本中作大量的修改。其他开发人员在读脚本时也很难理解脚本的实现方法，无法进行二次开发。由此可见，这种脚本是十分失败。在之后持续的编写中，考虑更多的是脚本中参数、函数的复用性，以及脚本的易读性，需要随时准备对脚本结构和函数进行重构。具体体现在如下几个方面：

（1）模块化。通过对脚本功能的分析，脚本可以分为数据抓取、数据处理、邮件发送、设置几个模块，最后在主函数中进行调用。

（2）提高代码复用性。通过分析各代码在脚本中的使用情况，当一部分功能代码在脚本中多次被使用，可以考虑将该部分代码封装成函数。这样在进行功能修改时只需要修改功能函数即可。

（3）将可修改参数集中。在脚本中常常存在一些参数需要根据需求进行修改，这些参数分布在脚本各模块的不同位置，在修改脚本时需要对所有的脚本模块进行修改，工作量较大且易遗漏修改。可以通过将所有可以修改的参数集中放在脚本某一位置或单独的脚本中，如本脚本中使用settings模块将所有参数集中在一起，其他模块都调用该模块。

（4）代码整洁美观。整洁美观的代码会让脚本的代码结构更加清晰，提升开发者的便携体验，造成代码混乱的往往是一些字符串，它们可能很长或存在特殊字符。可以通过将这些字符串映射为变量值，并将它们集中编写，这样在使用这些字符串是只需要直接调用这些变量即可。

（5）良好的注释。良好的注释并不是指注释越多越好，对于大多数简单的语句注释并没有意义，常需要注释的是参数、功能函数、类、关键功能语句。

（6）规范代码书写。 规范的代码书写能使再重新读代码时迅速了解代码的含义，代码规范体现在编写时的方方面面，如命名方式、注释方式、换行等等。使用Pycharm编写脚本时会对不规范的语法进行波浪线提醒，可以很好地对语法进行约束。

（7）方便调试。由于该脚本运行时间较长，在对新改动进行测试时需要等待脚本完全运行后才能知道结果。本脚本中通过在每个模块中添加if \_\_name\_\_ == '\_\_main\_\_':判断，可以运行单个模块进行单个模块的调试。

脚本在“每日构建”过程中也显露出了很多问题，目前脚本仍存在如下问题：

（1）运行时间较长。在脚本运行过程中发现，脚本运行的时间主要集中在抓取数据时的模拟点击和图片压缩。前者是由于使用selenium抓取数据是模拟点击时间间隔造成的，后者是由于在使用Tinify调用api过程较长引起的。目前考虑使用python中的多线程使多个webdrvier并行抓取数据，同时对前面生成的图片并行地压缩图片。

（2）缺少日志。虽然在Jenkins的构建中会将脚本构建情况进行保存，但是往往在脚本报错时这些提示是不够的，需要加入日志系统对脚本的各个阶段进行提示和记录。

## 本章小结

本章分析研究了基于Jenkins运行的VR生产环境的自动化脚本和VR应用测试情况自动报表脚本，实现了Jenkins服务器下的功能扩充。首先研究和分析了在VR生产环境中数据库自动化的必要性和在持续集成中的理论体现，再通过分析脚本需求对脚本实现的的代码对脚本的主要结构进行描述，对脚本运行结果进行分析。然后研究和分析了在项目管理中，持续集成中的报表环节的重要性，再通过分析需求对自动报表脚本的实现对脚本的主要结构进行描述，对脚本运行结果进行分析，并总结了编写脚本过程中的感受，侧面体现了持续集成的重要性。

# 总结与展望

# 致谢

# 参考文献

1. 惠特克. Google软件测试之道 : 像Google一样进行软件测试. 北京 : 人民邮电出版社, 2013.
2. 亨布尔. 持续交付 : 发布可靠软件的系统方法 : reliable software releases through bulid, test, and deployment automation. 北京 : 人民邮电出版社, 2011.
3. 贝克. 测试驱动开发 : 实战与模式解析 : by example. 北京 : 机械工业出版社, 2013
4. 巴顿. 软件测试 : 第2版. 北京 : 机械工业出版社, 2006.
5. 王宁. 基于Jenkins的持续集成系统的设计与实现[D]. 北京邮电大学, 2014.
6. 卞孟春. 基于Jenkins的持续集成方案设计与实现[D]. 中国科学院大学（工程管理与信息技术学院）, 2014.
7. 徐仕成. 持续集成在现代软件开发中的应用与研究[D]. 中南大学, 2007.
8. 徐怀宝, 肖悦娱. PC平台虚拟现实API自动化测试系统的研究与实现[J]. 电子测量技术, 2017(08):37-41+46.
9. 张兆鹏. Android平台的自动化测试系统的实现与优化[D]. 哈尔滨工业大学.