基于CI的Web自动化测试平台设计

李真辉[1],陈闻宇[1],徐彦之[2]

1. 中国互联网络信息中心,北京 100190
2. 中关村互联网金融行业协会,北京 100080

摘要 Web相关软件的开发更新快速、变化频繁、周期短,用户需求也变化多样,对软件测试的快速响应和质量提出了较高要求。为提高软件测试效率及准确率,设计了一套针对Web软件的自动化测试平台。该设计根据软件分层思想,基于持续集成平台Jenkins设计搭建,集成了Ant、TestNG、Selenium 2.0、Sonar等主流自动化产品,实现了软件系统的持续构建、持续测试、持续代码质量监控等的全流程测试自动化。该测试架构可将传统Web测试效率提高50%、准确率提高30%,同时严格控制了软件产品质量,降低了项目风险。

关键词 持续集成;自动化测试;代码质量;Selenium;Sonar

中图分类号 TP311 **文献标志码** A **doi** 10.3981/j.issn.1000-7857.2015.09.014

A solution to Web software automatic test based on continuous integration

LI Zhenhui[1], CHEN Wenyu[1], XU Yanzhi[2]

1. China Internet Network Information Center, Beijing 100190, China
2. Z-Park Association of Internet Finance, Beijing 100080, China

Abstract The web related software development shows a trend of more frequent changes in user demands and a shorter development period. In order to improve the efficiency and the accuracy of software test, a quick response high quality testing platform has been successfully built. According to the layered software architecture and based on the continuous integration platform, Ant, Selenium 2.0, TestNG, and Sonar are integrated to provide a solution to solve continuous construction, test and code quality monitor of the whole process of testing automation of a software system. It is shown that the solution can greatly improve software test efficiency with a high reliability.

Keywords continuous integration; automatic test; code quality; Selenium; Sonar

网络的高度普及使得软件产品系统包含越来越多的页面(Web)应用,相对于传统的应用,互联网Web应用具有产品调整快、系统变更频繁、海量用户、用户环境各异等特点[1],决定了Web测试不能简单地用传统测试方法。据统计,在早期Web应用开发过程中,采用人工测试所耗费的工作量占据了总开发工作量的60%。尤其是近年互联网用户对微博、维基、社交平台等互联网内容需求的增长,Web应用程序系统趋于结构更复杂、内容更庞大[2]。Web浏览器的种类日益多元化,使得对应的Web测试不仅需要检查和验证该Web系统是否按照设计的要求运行,而且需要测试应用在不同平台时浏览器下是否正常工作、正常显示及从用户角度进行可用性和安全性测试。

为了提高Web应用的测试效率、降低成本,国内外针对Web系统的自动化测试理论和测试架构进行了探索。基于行为驱动开发的自动化测试方法研究[3]主要针对Web前端测试,对静态测试有所欠缺;刘培[4]介绍的自动化测试平台设计,注重对自动化测试原理的阐述,但对具体实践提及较少,实际运用到项目测试中还需要大量的准备和预研;赵金丹[5]介绍的国内外流行的软件自动化测试工具使用则着眼于对相应软件深度的挖掘分析。目前国内对基于用户互动的

收稿日期:2014-09-12;修回日期:2014-12-08
基金项目:全球互联网名称与数字地址分配机构开源项目
作者简介:李真辉,工程师,研究方向为软件工程、测试自动化等,电子信箱:lizhenhui@cnnic.cn;陈闻宇(通信作者),工程师,研究方向为软件工程、项目管理,电子信箱:chenwenyu@cnnic.cn
引用格式:李真辉, 陈闻宇, 徐彦之. 基于CI的Web自动化测试平台设计[J]. 科技导报, 2015, 33(9): 78-82.

Web应用程序的测试大部分还处于测试平台和工具的简单应用,效率较低,没有真正意义上的能够普适各种Web开发的结论性成果。

本文分析日常软件测试实践工作中遇到的问题,结合软件测试的一般过程,提出一个有效的软件自动化测试解决方案。

1　CI自动化测试平台设计

1.1　测试平台总体设计

测试平台是自动化测试系统的基础,构建敏捷开发的自动化测试方案首先要考虑平台设计能够支撑全自动化测试的全流程,在此基础上综合考虑安全,快捷、准确、扩展性和自我定制等要求,测试平台设计至少应该包含如下因素。

1）源代码管理:支持目前广泛使用的集中式和分布式版本控制(如Subversion和Git)。

2）测试框架支持:如xUnit、TestNG、RobotFramework等通用的主流测试框架。

3）动态测试:同时支持功能测试和性能测试。

4）静态测试:支持对文档质量,代码的规范、安全、可靠、可维护,测试覆盖率等的自动化检查及量化分析。

5）部署相关:支持构建结果持续发布,便于快速发布软件到多种产品环境,支持发布至多种服务器。

6）结果展示:具有完善的报表功能和统计功能,且能够通过图形界面、表格、曲线等多种方式友好展示单次测试结果和历次变更趋势。

现已有较多的持续集成平台(Continuous Integration Server)能满足上述要求。从运作模式区分,CI平台有商业软件、开源软件两大类。其中Jenkins源自Hudson,并继承发扬了Hudson的诸多优点,功能完备,又因丰富的插件使得平台功能大为增强,目前已在很多行业的软件研发中广泛使用。基于测试平台安全、可扩展、成本、口碑和业界使用情况等多方面的考虑,采用Jenkins作为自动化测试平台具有较大的优势。

1.2　测试平台子模块设计

测试平台选定后,必须对平台进行相应设置,并规划好相应的子模块,才能确保顺利完成自动化测试,主要包含以下关键环节。

1）自身安全保证。应用系统处于网络之中,安全控制尤其重要。在Jenkins平台的系统管理中“启用安全”,把用户权限设置分为安全域(Security Realm)以及授权(Authorization)。在安全域,采用Jenkins内部数据库。如有更高的要求也可考虑Github托管、Servlet容器(Tomcat和GlassFish)接管、LDAP、Unix用户/组数据库等。在授权方面采用项目矩阵授权(Project-based),也可采用Github,安全矩阵(Matrix-based)或角色授权(Role-based)等策略。

2）持续构建(Continuous Build)。持续构建策略是保证自动化测试成功的基础因素,持续构建能够保证团队代码始终处于可联调,可编译的状态。建议团队代码因为安全原因需要隔离外,其他一律要求采用集体所有权。

3）持续测试(Continuous Test)。持续测试通过动态测试的方式实现,目的是检查主线上的代码是否能够实现新需求的功能,或者是否被破坏了已有功能。

4）持续代码质量控制(Continuous Inspection of Code Quality)。持续代码检查通过静态测试的方式实现,目的是保证代码风格一致。主要关注文档和代码的静态质量和内部质量及动态测试中单元测试质量。

1.3　子模块关联设计

ANT是基于JAVA的编译工具,通过调用ANT脚本(缺省名称build.xml)的target树,就可以执行各种任务(task)序列,每个任务实现了特定接口对象。Ant中的任务可以为3类。

1）核心任务。核心任务是Ant自带的任务。本文主要包含源代码的编译、Package(打包)、生成javadoc、清理部署临时文件等。

2）可选任务。可选任务是来自第三方的任务,通过附加的JAR文件实现。

3）用户自定义的任务。本文中主要用于实现远程Web服务控制以及war在服务器容器中的静态部署。

自动化测试平台以Jenkins CI平台为基础,以ANT编译工具为主线,实现Web系统的持续构建、持续测试、代码质量控制和持续部署的全流程自动化,实现了软件系统质量评估的全程量化管理(图1)。

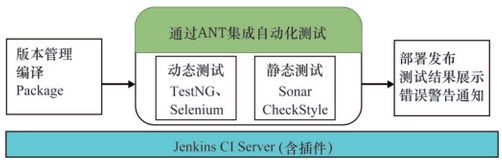

图1　基于CI的自动化测试平台示意

Fig. 1　Automatic test platform based on continuous integration

2　项目持续构建

项目持续构建贯穿了从项目启动到结束的全部测试周期,是设计自动化测试的源头。持续构建强调集成频率和及时反馈。目前推荐的最佳实践构建频率是每隔1 h构建1次,这也是后续测试和代码质量检查的频率。根据敏捷开发创始人之一Martin Fowler的观点,项目bug的增加与时间的平方成正比,两次集成间隔的时间越长,bug增加的数量越超预期,解决bug付出的工作量也越大。及时反馈测试结果(尤其是失败信息)尽早通知开发人员,使问题快速解决。

2.1　持续构建实现

构建被触发后,平台会从SCM检出源码,编译源码并生成Package,确保在任何时间、任何环境上都能够发布可以部署的软件。持续构建是Jenkins平台的基本功能,良好的构建能确保整个测试过程准确高效。图2为Jenkins平台持续构

建流程,很好地体现了持续构建的一些关键要素。

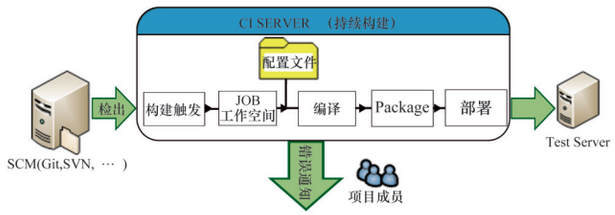

图2 Jenkins平台持续构建流程

Fig. 2 Continuous building process of Jenkins platform

1）构建触发。自动触发构建是自动化测试的基础,Jenkins构建触发主要包含3种模式。Poll SCM,定时检查源码,一旦有变更就构建。Build periodically,无论源码是否变更都周期性地构建。其他工程构建完毕后触发。

2）独立测试环境(工作空间)。将环境相应的配置文件保存在CI服务器上。不同环境所需数据库、参数配置不尽相同,源代码检出到JOB的工作空间(workspace)后,将CI上保存的配置文件覆盖开发环境配置,与其他代码一并进行编译、Package后部署,满足不同环境的实际需要。

3）编译-生成Package。系统源代码、单元测试代码和系统测试代码均应全部参与构建。构建过程中所产生的日志和package,可通过shell完成备份,以保证每个版本的发布全部可追溯。

2.2 构建后部署

持续构建完成后,待测试的软件代码就已经在平台经过编译,处理成可供测试的代码形式,但代码是否切实可靠有效还需要在预先设定的工作环境中进一步检测。根据软件在不同的研发阶段,构建后的部署需要针对不同的工作环境,主要包含开发环境、测试环境、用户验收测试(UAT)环境、正式运营(Product)环境。

每次构建都需要部署到开发和测试环境,部署频率最高,因此部署也是一个持续的开发实践方法,用来确保把代码快速、安全地部署到产品环境中。每次改动代码都被提交到模拟产品环境中,经过严格的自动化测试,确保业务应用和服务符合预期。

每个变更被自动提交到测试环境中并得到充分测试验证,所以当软件系统开发完成,业务条件具备时,只需按一次钮,就能将应用系统安全地部署到UAT或者正式运营环境。

本文的构建是在平台中启用SCP Publisher或Deploy插件实现远程动态部署;该平台也可以在ANT的build.xml文件中,通过SSHEXEC Command人为控制远程服务的启停,执行SCP shell命令拷贝安装包,从而实现静态部署。

2.3 测试结果快速反馈

为确保构建正常,一旦构建过程发生错误,就必须尽快通知相关人员,促使问题快速修复。针对2种类别的信息,自动化测试平台在一些关键点对测试结果做出了判断:

1）构建错误。如一旦发生源代码编译错误、Package部署错误、系统测试错误等,自动化测试流程就可能完全停止(错误级别较高)。

2）测试结果未达预期。如测试覆盖率、文档注释率、单元测试通过率等未达到设定的阈值时,不必终止自动化测试流程(错误级别较低)。

在自动化测试平台设计中,不同级别的错误区别对待,采用不同的消息通知机制。对于级别较高的错误,采用积极方式主动推送(如弹出窗口、Chat、短信等)。而对于级别较低的错误,采用消极方式进行发布(如邮件、RSS消息,需要相关人员去阅读)。自动化测试平台的消息通知机制使得平台的高效得以真正的体现。

3 项目持续测试

完成项目源代码的持续构建和部署后,需要在部署的环境上对被测系统实现持续自动化测试,即项目持续测试。API的自动化测试较为好开展,而Web系统则实施难度较大。Web系统的自动化测试工具较多,早期QTP和LoadRunner等采用“录制-回放”脚本的方式,对环境的依赖性太强,对Web页面的变化过于敏感,后期脚本的维护工作量很大,无法满足敏捷开发的需要且不能与Jenkins等CI平台集成。在Jenkins平台中利用采用Ant编译工具执行第三方任务,引入自动化测试框架TestNG和Selenium测试工具,实现Web系统自动化测试。

3.1 Selenium自动化测试

Selenium 2综合了Selenium1.0和Web Driver各自的优点,Selenium 2在Web自动化测解决了Selenium 1.0存在的JS沙箱问题,支持更广泛的浏览器和编程语言,并提供非常易用、可读性很强的API。综合第三方的因素,当前Selenium支持的浏览器包含桌面浏览器(Firefox、InternetExplore、Opera、Safari 和Chrome),移动平台浏览器(Android、IOS和Windows Phone),类浏览器(HtmlUnitDriver和PhantomJS)[11]。

3.2 Selenium与Jenkins集成

为了将Selenium集成到测试平台中,引入TestNG框架。通过Ant调用TestNG框架,然后在TestNG框架中引入Selenium 2.0插件,使Selenium的测试方法在Jenkins平台中执行。测试结果也可通过插件得到很好展示。TestNG测试框架如图3所示。

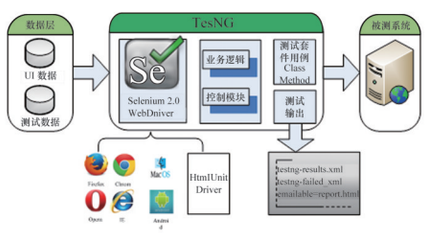

图3 TestNG测试框架

Fig. 3 TestNG framework

在TestNG框架中连接Selenium插件需要额外添加一些定义文件,以确保Selenium能够在TestNG中正确工作。Sele-

nium作为一个较为独立的功能插件,并未专门为TestNG框架设计专有嵌合功能,所以需要单独编写对应的链接设置文件。具体操作步骤如下:

1)在build.xml中使用<taskdef>定义第三方任务TestNG ant task:

<taskdef resource="testngtasks" classpath="./AntBuild/ /lib/testng-6.8.jar" />

2)指定class路径,测试结果输出路径以及test suite文件:

```
<path id="cp">
<pathelement location="./AntBuild/warsrc/WEB-INF/classes" />
<pathelement location="./AntBuild/warsrc/WEB- INF/lib/selenium-java-2.40.0.jar"/>
</path>
<target name="autotest" depends="build">
  <testng classpathref="cp" outputDir="${report}">
    <xmlfileset dir="${suite}" includes="testng.xml"/>
  </testng>
</target>
```

3.3 浏览器参数测试

测试验证需要在不同浏览器下进行多次简单的重复,Web系统需要兼容不同的浏览器。Selenium支持多种浏览器,TestNG支持参数化测试,二者结合,Web系统测试工作量数倍降低。验证系统对不同浏览器的兼容性验证步骤如下:

1)testng.xml的Test Suite中指定传入参数值

<parameter name="browser" value=" Chrome " />

2)在测试class中通过@ Parameters引入参数,采用Selenium WebDrive的测试方法。

4 代码质量控制

通过Selenium和TestNG,在Jenkins平台实现了Web系统的动态持续测试(功能性测试)。要实现软件系统的静态测试(代码质量验证),需要分别关联Checkstyle,,PMD, FindBugs,Cobertura、Clover和Jcoco等多种工具,配置十分繁琐。

本文在所搭建的测试平台中引入SonaQube平台以实现Web软件的代码质量控制。SonarQube很好的整合了上述工具的各种功能,并将以上工具的各种测试结果以多维度形式合并分析输出。作为目前最强大的代码质量管理平台之一,SonarQube支持包括java、C#、C/C++、Cobol、ABAP等20多种编程语言的技术分析和度量,在通信、金融、IT和制造业均有成功的应用(如AT&T、美国银行、ORACLE、FIAT)。

4.1 Sonar质量控制体系

SonarQube整合了多种静态测试工具,并基于SQALE(Software Quality Assessment based on Lifecycle Expectations)等软件质量方法理论,对多项检测结果进行再加工处理,便于对不同规模工程的代码质量进行量化评估[12]。

度量是Sonar的核心功能之一,SonarQube输出的是一个综合质量管理报告,其数据来源于7个维度的代码质量检测结果,代码检测维度,按重要性依次排序如下[13]。

1)重复度:展示代码行或代码块重复严重的地方。

2)复杂度:评估文件(File)、类(Class)、方法(Method)等的复杂度。复杂度过高将导致bug修复成本较高。

3)单元测试质量:获得行覆盖率和分支覆盖率,统计并展示单元测试覆盖率。

4)代码规范:通过CheckStyle、PMD、Findbugs等工具规范代码编写。

5)潜在bug:sonar可以通过PMD、CheckStyle、Findbugs等等代码规则检测工具检测出潜在的bug。

6)注释比例:尤其是关注Public API的注释。

7)架构和设计:通过Sonar找出循环,展示包与包、类与类之间的相互依赖关系。

4.2 Sonar平台部署

Sonar平台有自己独立的用户、组和权限管理,Jenkins服务器能够通过JDBC连接到Sonar的数据库。在持续构建的同时把代码质量检查结果直接写入Sonar DB,供Sonar Web Server读取,展现给用户。

Sonar默认安装情况下使用其自带数据库Apache Derby,并支持数据库,包含MySQL 5.x、Oracle thin模式(含10g,11g和XE)、Postgresql、MS SqlServer。图4是Sonar代码质量控制平台。

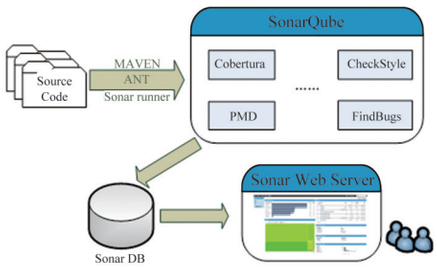

图4 Sonar代码质量控制平台

Fig. 4 Sonar coding quality control platform

4.3 Sonar与Jenkins集成

Jenkins和Sonar是两套独立的服务系统,首先需要在Jenkins平台Ant的build.xml文件中定义Sonar DB属性,代码如下:

```
<property name="sonar.jdbc.url" value="jdbc:oracle:thin:@218.241.1.3:1521:qadb" />
<property name="sonar.jdbc.username" value="sonar" />
<property name="sonar.jdbc.password" value="sonar" />
<property name="sonar.host.url" value="http://boss180.sonar.cn:8600"/>
```

其次,需要在build.xml中定义Sonar项目属性。

```
<property name="sonar.projectKey" value="org.cnnicRD:${name}" />
```

```
<property name="sonar.projectName" value="${name}" />
<property name="sonar.projectVersion" value="1.0" />
<property name="sonar.language" value="java" />
<property name="sonar.sources" value=" ${basedir}/src/main/java" />
<property name="sonar.binaries" value=" ${basedir}/Ant-Build/war" />
```

然后,通过taskdef定义第三方任务,调用启动Sonar,并将质量代码检查结果入库供Sonar Web Server查询使用。

```
<target name="sonar" depends="autotest">
<taskdef uri="antlib:org.sonar.ant" resource="org/sonar/ant/antlib.xml">
<classpath path="/home/Jenkins/.jenkins/sonar/sonar- ant-task-2.1.jar" />
</taskdef>
<sonar:sonar />
</target>
```

5 平台实际工作验证

测试方案在ICANN(The Internet Corporation for Assigned Names and Numbers)的RestFul WHOIS项目中投入实际运用,项目每天9:00自行启动持续集成,另根据代码更新等情况人为择机启动。整个测试工作除例行监控外,基本脱离了人工测试,测试效率获得了极大的提高。按照项目预计评估,整个测试按照传统Web测试(测试脚本运行加部分人工)需要大约2人·月工作量,使用本架构进行自动化测试后实际测试工作量为7人·天。自动化测试结果经人工复核无误报。整个测试过程分为动态测试及代码静态测试两部分。

动态测试部分:项目随着敏捷开发推进,测试用例数量逐渐增大,这一特点使本平台能在测试过程中在项目中很好的践行持续集成、持续测试、持续可用的设计理念。图5是项目中历次自动化测试执行的趋势。

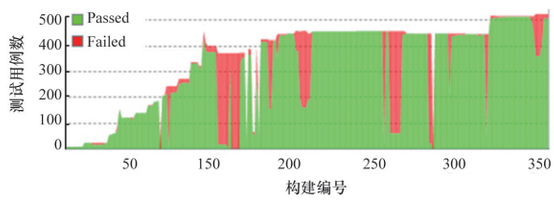

图5 TestNG测试结果趋势图

Fig. 5 TestNG results pass/fail trend

项目自动化测试获得主要数据如下:

1)项目共计build 351次,最高测试用例524个。

2)测试用例数量的变化趋势及历次构建中用例执行成功和执行失败的比例。

代码静态测试部分:代码质量的检查结果在sonar的项目仪表盘中统一展示,能够准确地把问题定位到源代码文件的具体位置,代码修改快捷。项目代码质量检查结果为:

1)重复度:度量结果为4.5%。体现在20个文件,1,362行,44块中。

2)复杂度:度量结果为2096。在方法、类和文件中的平均复杂度分别为2.8/方法,9.9/类,10.5/文件。

3)包耦合指数:5.4%,循环>13。

4)文档规范:公共API 506个,未注释API 26个,API注释达94.9%。

5)单元测试覆盖:代码覆盖率68.5%,行覆盖率70.8%,分支覆盖率61.0%。

6)单元测试执行:合计执行218次。0失败,0错误,成功率100%。

7)问题:出现主要问题174次,未出现阻断错误、严重错误、次要错误和信息错误。完成全部问题修改的技术债务为7.6天。

6 结论

对软件测试过程进行深入分析,提出了一个有效可行的解决方案,实现了Web测试全流程自动化。本设计较目前国内主流测试平台更为全面,具有更好的通用性,尤其在架构设计上对多种自动化测试平台的功能挖掘更深入。经试验,采用本文设计的测试架构可将传统Web测试效率提高50%。基于此架构的浏览器类型参数化,将回归测试效率更是提高数倍,准确率提高30%。极大提高测试效率同时,严格控制了软件产品质量,降低项目风险。

参考文献(References)

[1] 许蕾, 徐宝文, 陈振强. Web测试综述[J]. 计算机科学, 2003, 30(3): 100-104.
Xu Lei, Xu Baowen, Chen Zhenqiang. A survey of Web testing[J]. Computer Science, 2003, 30(3): 100-104.

[2] Xu Lei, Xu Baowen, Jiang Jixiang. Testing web applications focusing on their specialties[J]. ACM Special Interest Group on Software Engineering, 2005, 30(1): 1-6.

[3] 曹洋, 崔萌. 基于行为驱动开发的自动化测试方法研究[J]. 清远职业技术学院学报, 2013(6): 1-4.
Cao Yang, Cui Meng. The research on behavior-driven development of automated testing [J]. Journal of Qingyuan Polytechnic, 2013(6): 1-4.

[4] 刘培. 自动化测试平台的设计与实现[J]. 科技创新与应用, 2014(24): 45-48.
Liu Pei. The platform of automated testing[J]. Technology Innovation and Application, 2014(24): 45-48.

[5] 赵金丹. 基于selenium的web自动化测试脚本设计研究[J]. 科技传播, 2014(1): 94-95.
Zhao Jindan. The research on selenium of web automated script testing [J]. Public Communication of Science & Technology, 2014(1): 94-95.

[6] Smart J F. Jenkins: The definitive guide[M]. Sebastopol: O'Reilly Media, 2011.

[7] Moser M, O'Brien T. The hudson book[M]. Redwood Shores: Oracle, Inc., 2011.

[8] Beust C, Suleiman H. Next generation java testing[M]. Boston: Addison-Wesley Professional, 2007.

[9] Menon V. Test NG Beginner's guide[M]. Birmingham: Packt Publishing, 2013.

[10] Eric M B, Tilly J. Ant:The definitive guide[M]. Sebastopol: O'Reilly Media, 2002.

[11] 赵卓. Selenium自动化测试指南[M]. 北京:人民邮电出版社, 2013.
Zhao Zhuo. Automated Web testing with selenium[M]. Beijing: Posts & Telecom Press, 2013.

[12] Arapidis C. Sonar code quality testing essentials[M]. Birmingham: Packt Publishing, 2012.

[13] Racodon D. Developers' seven deadly sins [EB/OL]. (2013-12-03) [2014-01-06]. http://docs.codehaus.org/display/SONAR/Developers%27+Seven+Deadly+Sins.

(责任编辑 赵业玲)