## 主题3-如何自动验证tomcat各个模块的功能

关于自动验证Tomcat的功能，我们还是非常感兴趣的。后续我们在持续优化Tomcat过程中，会通过Tomcat release的机制，不断发布新的版本。那么，我们如何保证版本的可靠性呢？

我们知道Tomcat的功能是非常多的，如果要依靠手工验证，显然是不可能的。所以，我们要参考TestUI框架，搞一个自动化验证的工具。能够自动对tomcat本身的存量功能进行回归验证。

# tomcat官网的单元测试

我们知道自动化测试分为好几个层次：

1. UI测试
2. 接口测试
3. 单元测试

对于单元测试，一般成熟的开源工程，都会包含test package，对各个模块进行测试。tomcat当然也不例外，我们看tomcat官网github工程也有test package:

|  |
| --- |
| github官方源码库：  <https://github.com/apache/tomcat>  我们下载到了本地(iMac home路径)  /Users/zhoushuo/Documents/workspace/testGit/tomcat |

# 单元测试1-TestRemoteIpValve

我们随便找一个例子。比如有人开发了一个valve：RemoteIpValve。这个valve的功能是能够记录客户端的IP，并且把remote clientip放到request http header中。那么，怎么验证RemoteIpValve的功能呢？下面这个案例就说明了如何通过JUnit对RemoteIpValve进行单元测试。

## 代码示例

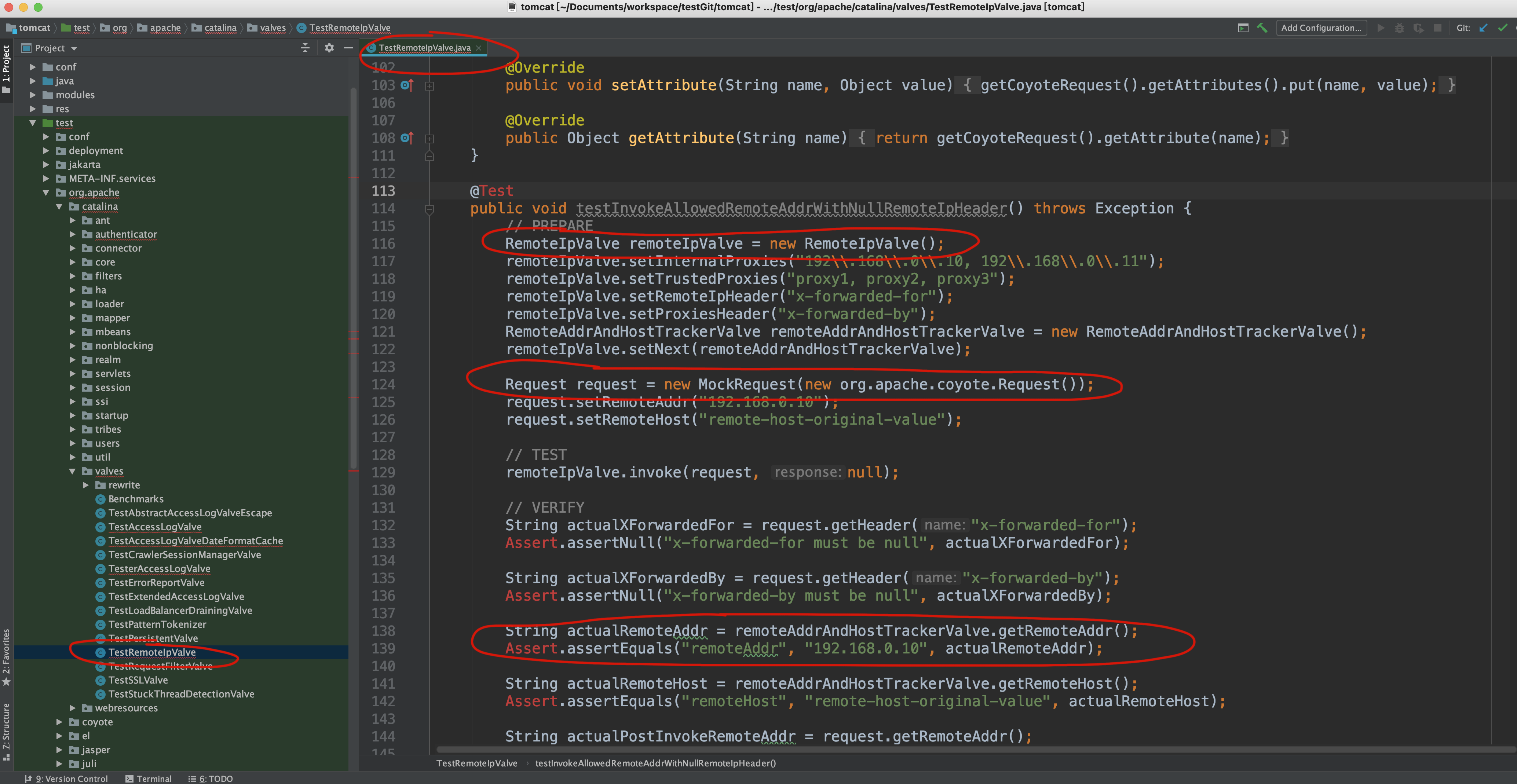


图1 验证RemoteIpValve的JUnit单元测试案例

## 详细测试步骤

详细步骤如下：

1. 新建JUnit测试类：

|  |
| --- |
| JUnit测试类代码：  TestRemoteIpValve.java  所属的package:  test/org.apache.catalina.valves |

1. 新建RemoteIpValve实例

|  |
| --- |
| RemoteIpValve remoteIpValve = new RemoteIpValve(); |

1. mock一个Request实例对象

|  |
| --- |
| Request request = new MockRequest(new org.apache.coyote.Request()); |

1. 模拟RemoteIpValve实例处理request

|  |
| --- |
| remoteIpValve.invoke(request, null); |

5.通过asser验证valve处理结果

|  |
| --- |
| String actualRemoteAddr = remoteAddrAndHostTrackerValve.getRemoteAddr();  Assert.assertEquals("remoteAddr", "192.168.0.10", actualRemoteAddr); |

## 总结

这个例子说明了如何通过单元测试验证某个功能。

# 单元测试2-TestStandardContext

还有一种情况，就是我们知道tomcat在不断演进的同时，在不断地解决各种BUG。如何验证BUG已经被解决，如何保证tomat版本升级后，之前发生过的BUG不会重现？这些都需要单元测试来解决。

## 代码示例

比如下图的代码示例，就展示了如何通过JUnit单元测试验证某个之前出现过的BUG是否已经修复：

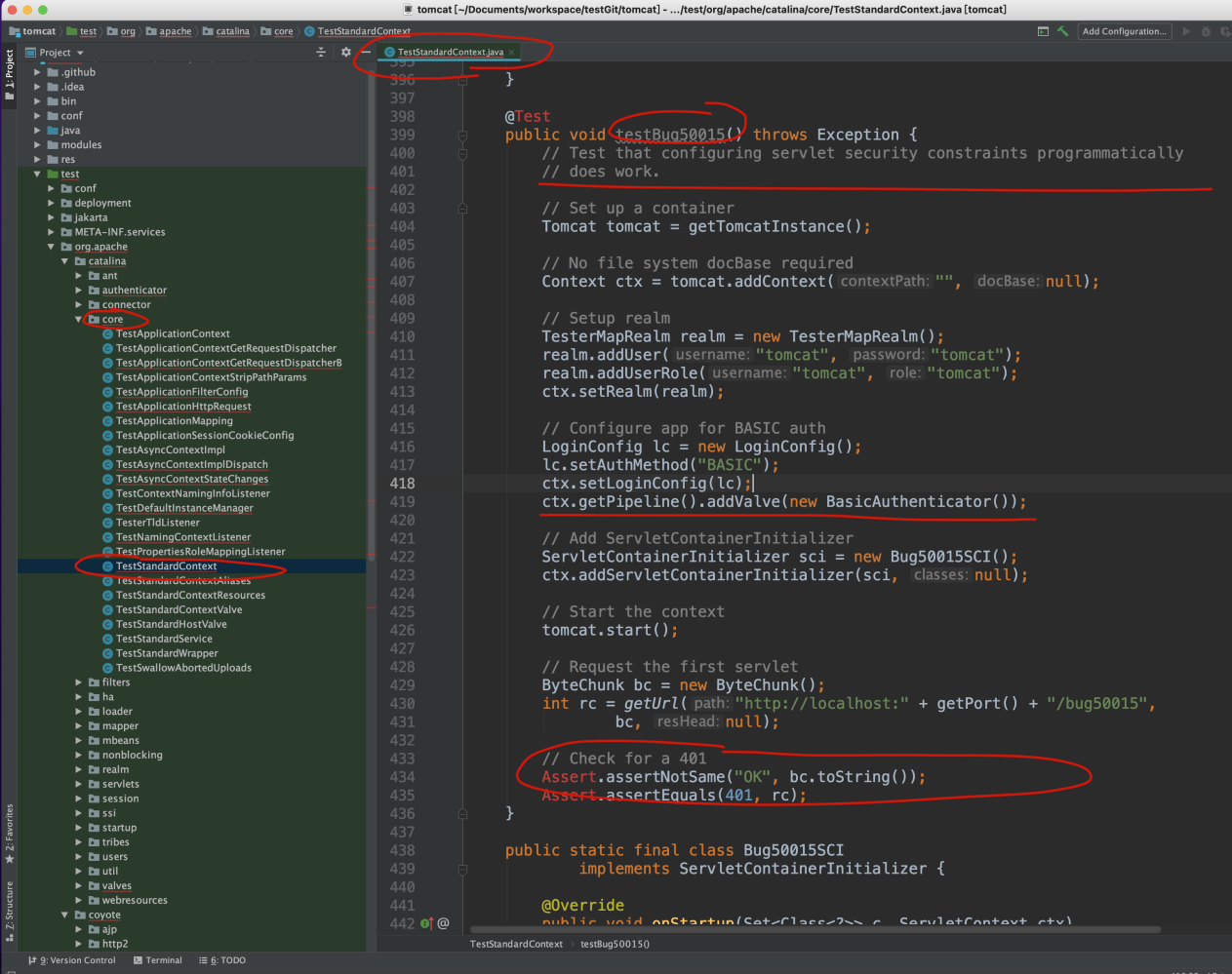


图2 TestStandardContext.testBug50015()验证某个BUG的解决情况

## 详细测试步骤

### 1.新建JUnit测试类

|  |
| --- |
| JUnit测试类代码：  TestStandardContext.java  所属的package:  test/org.apache.catalina.core  单元测试方法：  testBug50015() |

### 2.添加注释

|  |
| --- |
| // Test that configuring servlet security constraints programmatically  // does work. |

这段注释说得很清楚，这个测试方法用于验证某个已解决的BUG(Bug50015)，意思是我们配置了安全相关的配置，需要生效。我们猜测Bug50015是之前配置了安全相关的内容，但是因为某些原因没有生效。

### 3.建立tomcat实例

|  |
| --- |
| // Set up a container  Tomcat tomcat = getTomcatInstance();  // No file system docBase required  Context ctx = tomcat.addContext("", null); |

### 4.配置安全相关的内容

比如配置Realm(用户信息)、LoginConfig(登录信息)、设置authenticator

|  |
| --- |
| // Setup realm  TesterMapRealm realm = new TesterMapRealm();  realm.addUser("tomcat", "tomcat");  realm.addUserRole("tomcat", "tomcat");  ctx.setRealm(realm);  // Configure app for BASIC auth  LoginConfig lc = new LoginConfig();  lc.setAuthMethod("BASIC");  ctx.setLoginConfig(lc);  ctx.getPipeline().addValve(new BasicAuthenticator()); |

### 5.模拟servlet实例

tomcat实例有了，那么我们要请求的servlet如何配置呢？下面这段代码，就是通过创建一个Bug50015SCI类，实现了ServletContainerInitializer，来模拟一个servlet实例。

|  |
| --- |
| // Add ServletContainerInitializer  ServletContainerInitializer sci = new Bug50015SCI();  ctx.addServletContainerInitializer(sci, null); |

### 6.模拟tomcat启动，并发起request请求

|  |
| --- |
| // Start the context  tomcat.start();  // Request the first servlet  ByteChunk bc = new ByteChunk();  int rc = getUrl("http://localhost:" + getPort() + "/bug50015", bc, null); |

### 7.通过Assert验证结果

其实就是验证在tomat配置了安全相关信息的情况下，如果在servlet请求中没有带上用户信息，servlet请求就报错：

|  |
| --- |
| // Check for a 401  Assert.assertNotSame("OK", bc.toString());  Assert.assertEquals(401, rc); |

## 总结

这个单元测试主要是为了验证BUG的解决情况。

这一个短短的单元测试案例，最初目标当然是为了验证BUG50015S是否真正解决了。同时，这个测试案例包含的信息量还是很大的；

1. 这个测试案例包含了security的整体使用方法。我们如果对于security还有使用上的疑问，可以参考这个案例；
2. 这个案例包含了tomcat实例创建，servlet实例创建这些方法，对于我们自测非常有帮助。后续我们开发了一个新的功能，想要验证这个功能是否可用，可以借鉴这些测试案例。

# 后续

## 实际运行一下test类

tomcat官网的这些test类，有很多可以挖掘的地方，比如对于各种功能、场景的验证方式；比如test类中包含了各个核心组件的功能，我们实际跑一下这些案例，通过这些案例了解tomcat各个核心组件的功能。

## 开发test类

参考tomcat官网github上相关的test package，开发一个test类，对某个功能进行单元测试。

## Mock

了解tomcat单元测试各种mock技术，比如如何创建tomcat实例/servlet实例(参考单元测试2-TestStandardContext)、如何Mock Request(参考单元测试1-TestRemoteIpValve)、如何模拟http请求，等等。

# 参考

## 1.tomcat官网

tomcat官网github上，有相关的test package

## 2.mysql官网

同时，我们可以参考MySQL官网上的一个Test Framework

了解像MySQL这样的大型开源软件是如何通过测试保证软件质量的

https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE\_MYSQL\_TEST\_RUN.html

# 总结