针对保险核心业务系统的敏捷测试方案

吴穹
adwu73@hotmail.com
13701021050

残酷的现实

- 开发人员在巨大的业务压力下，每天只顾堆代码

残酷的现实

- 测试在Deadline前一天拿到代码，开始测试；测试到最后一份钟，为了和谐，发布结果为“通过”的测试报告

残酷的现实

- 系统部署生产环境后，出现严重问题，大家加班熬夜到凌晨，系统勉强上线

残酷的现实

- 第二天下午来上班，又开始第二个恶性循环

浪费在哪里？

- 反馈太慢，效率太低
  - 开发环境与Staging环境差异很大，许多问题要上生产环境才能发现；
  - 因此，测试被要求在Staging环境下进行；
  - 部署到Staging环境**最快**要花三个小时；
  - 许多简单的逻辑错误要到后期才在Staging环境下通过看日志的方式来发现；

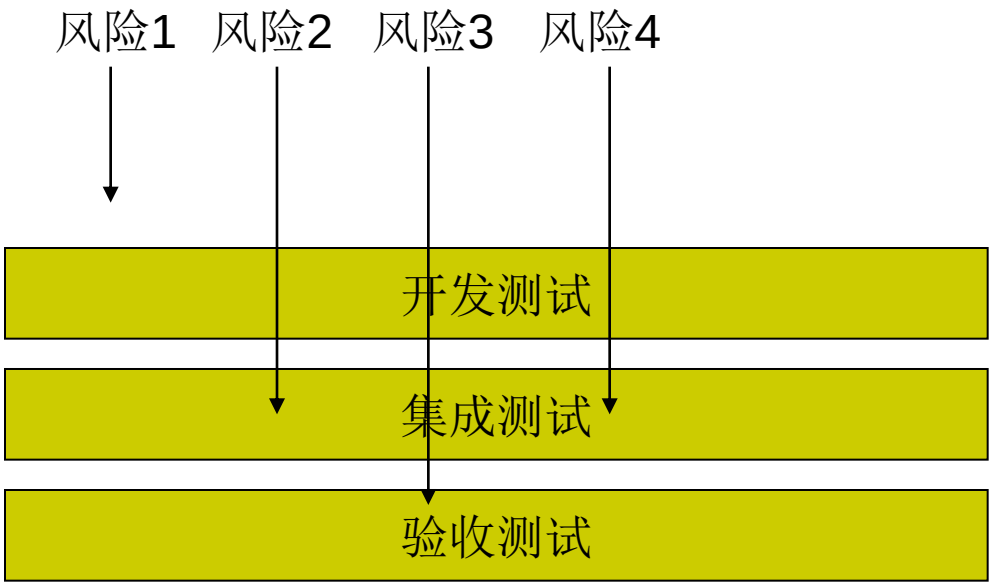
观点

- 快速反馈是敏捷的基础；
- 测试自动化又是快速反馈的基础之一；
- 测试自动化是一项繁重的工程，因此，我们需要巧妙地规划，决不能蛮干
  - 我们需要定义分层测试策略

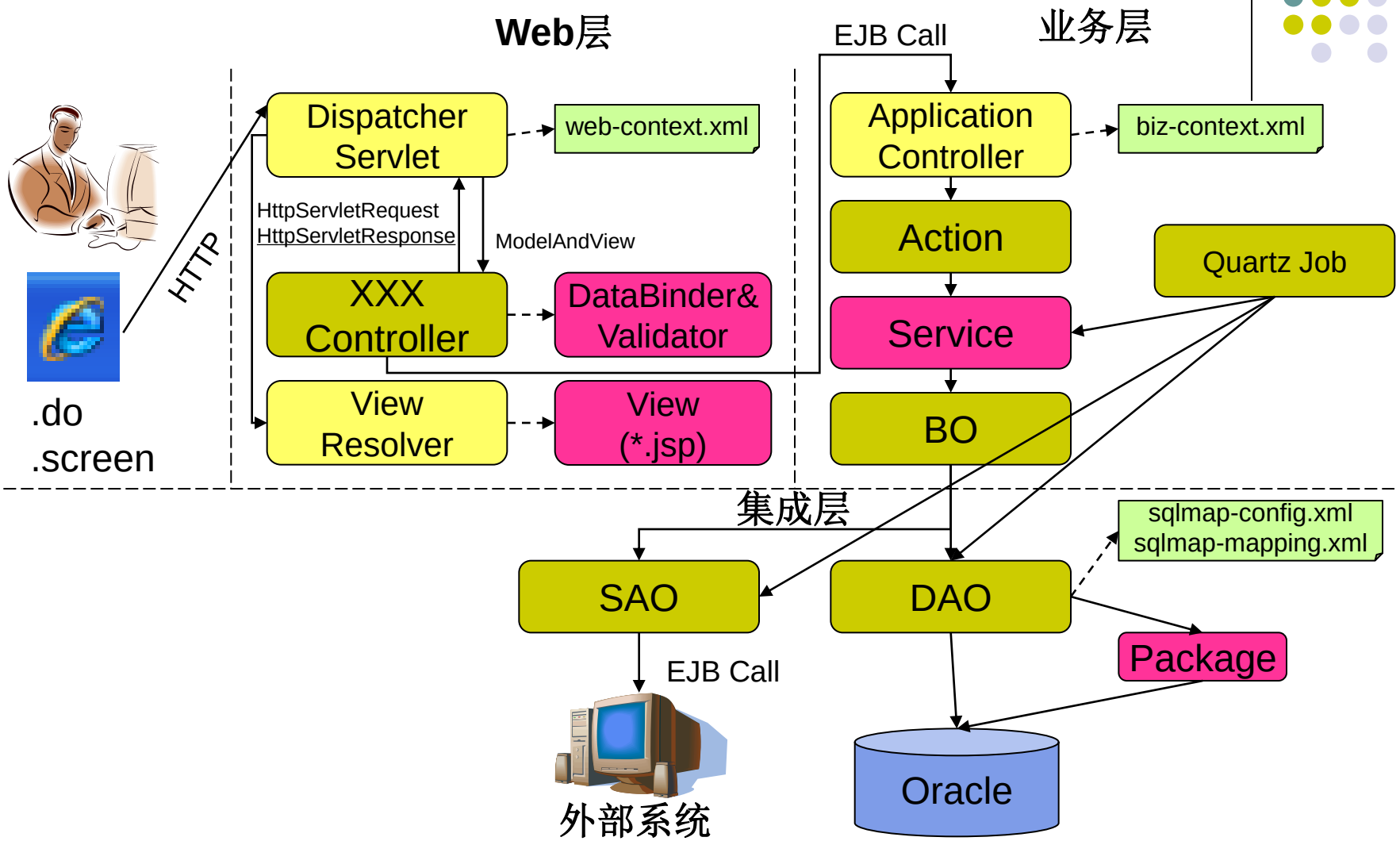
敏捷测试指导原则

- 场景是不可穷举的，测试活动必须是风险驱动的，关注于高风险的场景；
- 在正确的位置（层次）进行恰当的测试是自动化的关键；
- 仅靠测试人员不可能获得高质量的软件，质量是整个研发团队的责任；

分层测试的净水器模型



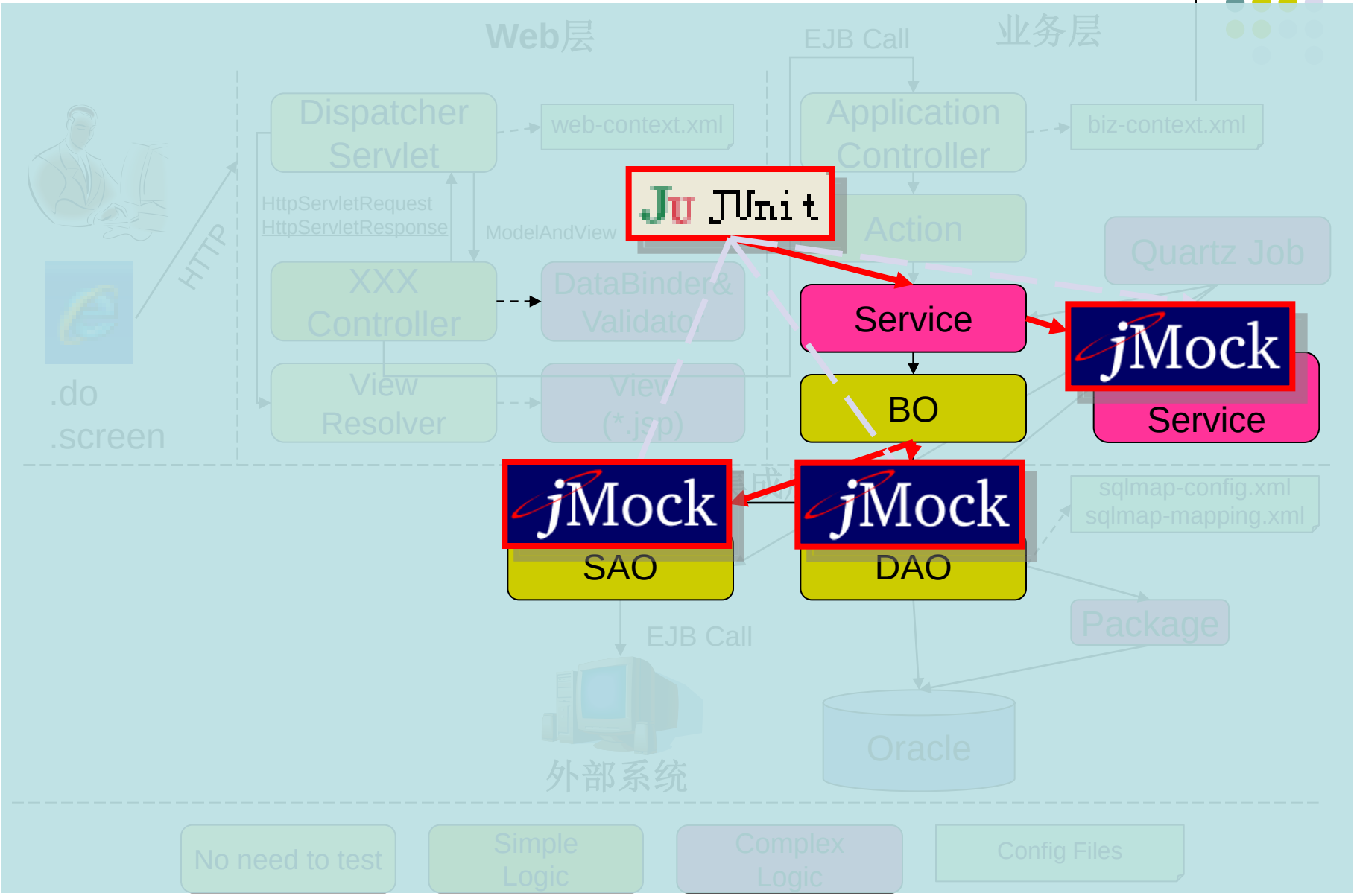
风险分析


Web层
业务层
EJB Call
Dispatcher Servlet
web-context.xml
Application Controller
biz-context.xml
HTTP
HttpServletRequest
HttpServletResponse
ModelAndView
Action
Quartz Job
XXX Controller
DataBinder& Validator
Service
View Resolver
View (*.jsp)
BO
.do
.screen
集成层
sqlmap-config.xml
sqlmap-mapping.xml
SAO
DAO
Package
EJB Call
Oracle
外部系统
No need to test
Simple Logic
Complex Logic
Config Files

分层测试策略

	开发者测试（DT）	控制驱动集成测试（CDIT）	界面驱动集成测试（IDIT）	验收测试（AT）性能测试
测试粒度	Service, Package, Job,…	子系统	业务领域	多个业务领域
关注的风险	一个模块内的逻辑功能错误	多个模块的集成问题，某些系统集成配置.	多个子系统的集成问题，某些系统集成配置，界面展示	关键业务功能的可用性，系统性能，可靠性
执行环境	开发者本机	开发集成测试环境	开发集成测试环境	Staging环境
测试类型	Out-Container	In-Container	In-Container	In-Container
数据库	N/A	测试数据	测试数据	类生产数据

针对Service/BO的开发者测试
（方案一、 MockDB方案）



针对Service/BO的开发者测试（方案一、 MockDB方案）

*XXXServiceTest.java

```
package test;

import java.util.List;


public class XXXServiceTest extends AbstractDependencyInjectionSpringContextTests {
    // this instance will be (automatically) dependency injected
    protected DVDInfoDao dvdInfoDao;

    public XXXServiceTest() {
        //Create Mock
        Mockery context = new Mockery();
        dvdInfoDao = context.mock(DVDInfoDao.class);

        ....

        //Set up expectations
        context.checking(new Expectations() {{
            oneOf (dvdInfoDao).getDVDByID(1);
            will(returnValue((list));
        }});

    }

    // specifies the Spring configuration to load for this test fixture
    protected String[] getConfigLocations() {
        return new String[] { "" };
    }

    public void testXXXService() throws Exception {

        ...

    }
}
```

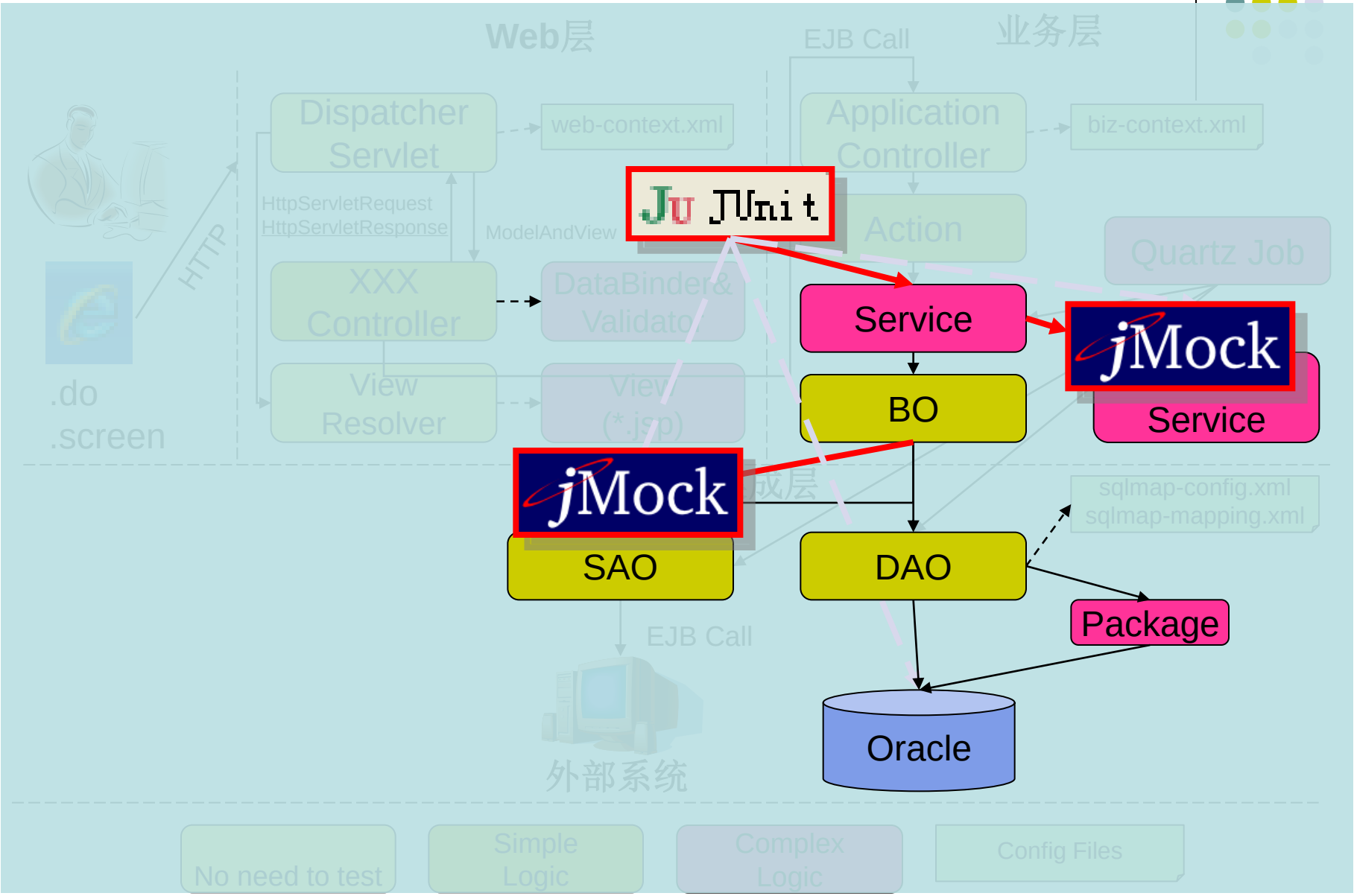
Finished after 0.735 seconds

Runs: 1/1 Errors: 0 Failures: 0

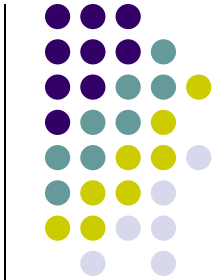
test.XXXServiceTest [Runner: JUnit 4]
testgetDVDInfoById (0.719 s)

Failure Trace

针对Service/BO的开发者测试
（方案二、 FeedDB方案）



针对Service/BO的开发者测试（方案二、 FeedDB方案）



*XXXServiceTest.java

```java
package test;

import java.util.List;


public class XXXServiceTest extends AbstractDependencyInjectionSpringContextTests {
    // this instance will be (automatically) dependency injected
    protected DVDInfoDao dvdInfoDao;

    public XXXServiceTest() {
        // switch on field level injection
        setPopulateProtectedVariables(true);
    }

    // specifies the Spring configuration to load for this test fixture
    protected String[] getConfigLocations() {
        return new String[] { "classpath:config/mydvd-data.xml" };
    }

    public void testXXXService() throws Exception {

        ...

    }
}
```

Finished after 0.735 seconds

Runs: 1/1 Errors: 0 Failures: 0

test.XXXServiceTest [Runner: JUnit 4]
testgetDVDInfoById (0.719 s)

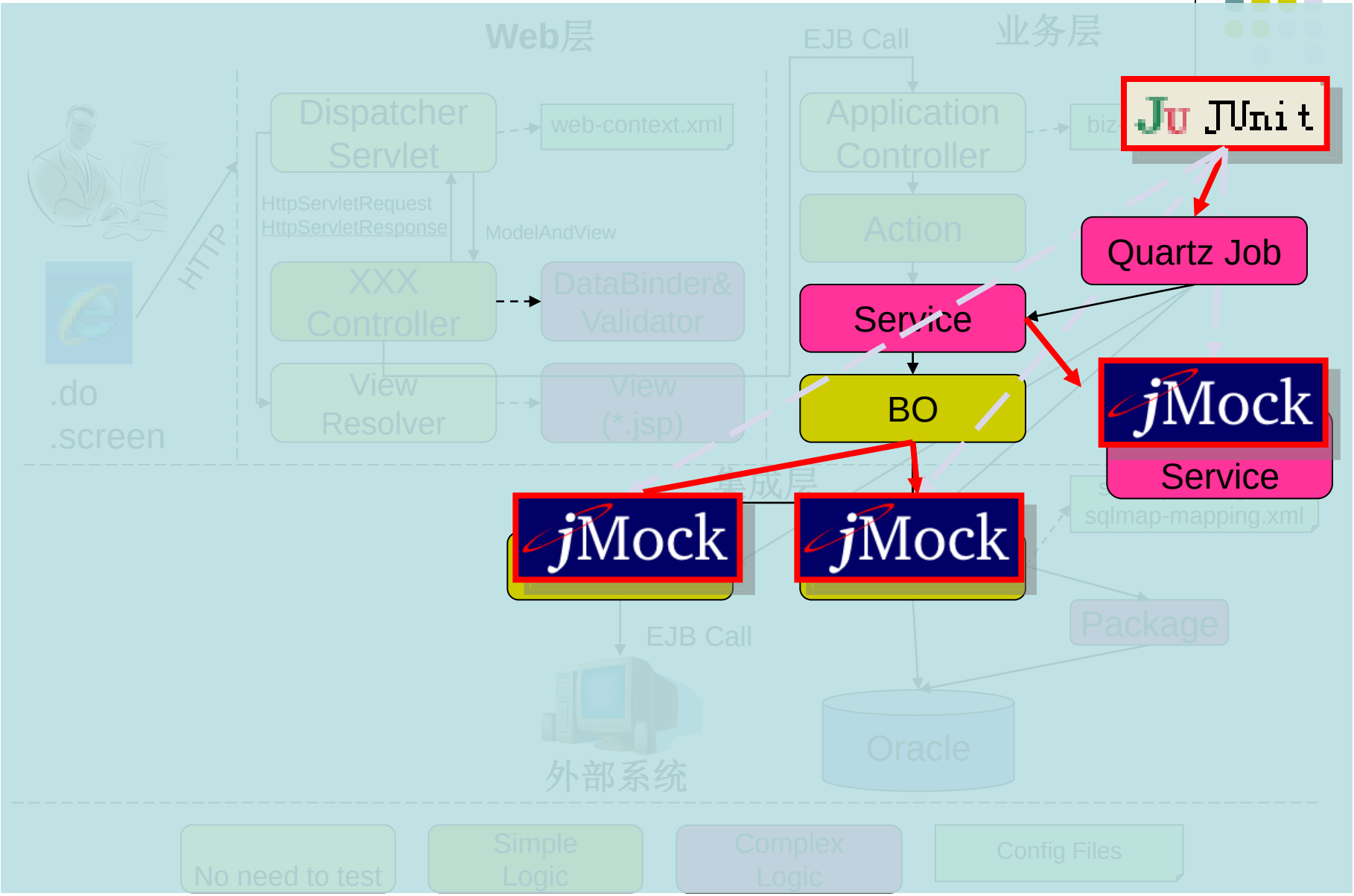
Failure Trace

针对Service/BO的开发者测试

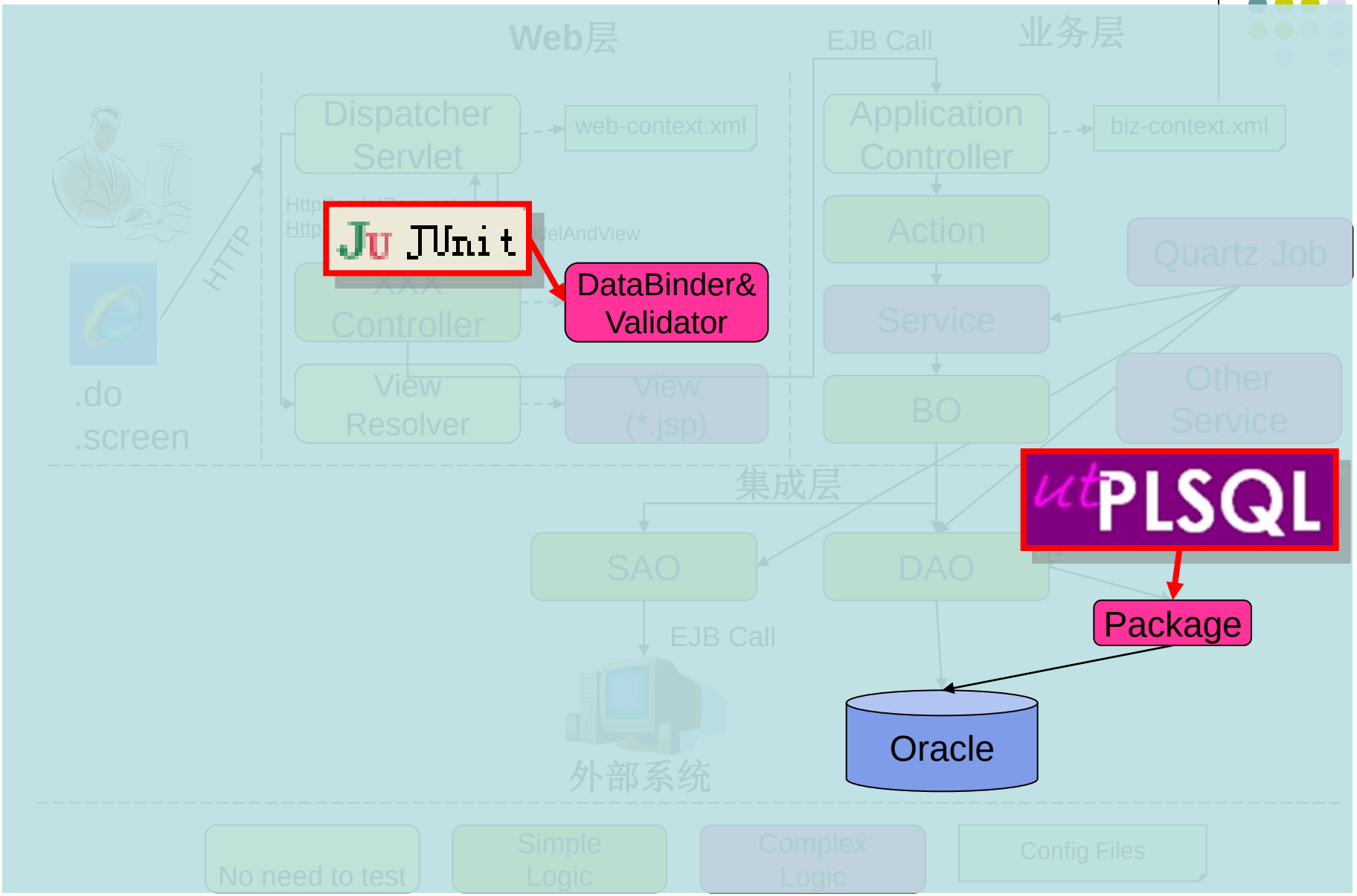
- 两个方案的比较：

	MockDB	FeedDB
开发人员工作量	相对较大，测试数据需要以JUnit测试用例来表达	相对较小，测试数据可以以Excel形式准备，便于复用
环境要求	没有	要求每个开发人员有一个数据库沙箱（可以是内存数据库，如hsqldb)
测试速度	较快	较慢
测试范围	较小	较大（可以测试iBatis xml配置文件，可以覆盖部分相关Package代码）

针对Quaztz Job的开发者测试
（同样有两个方案）



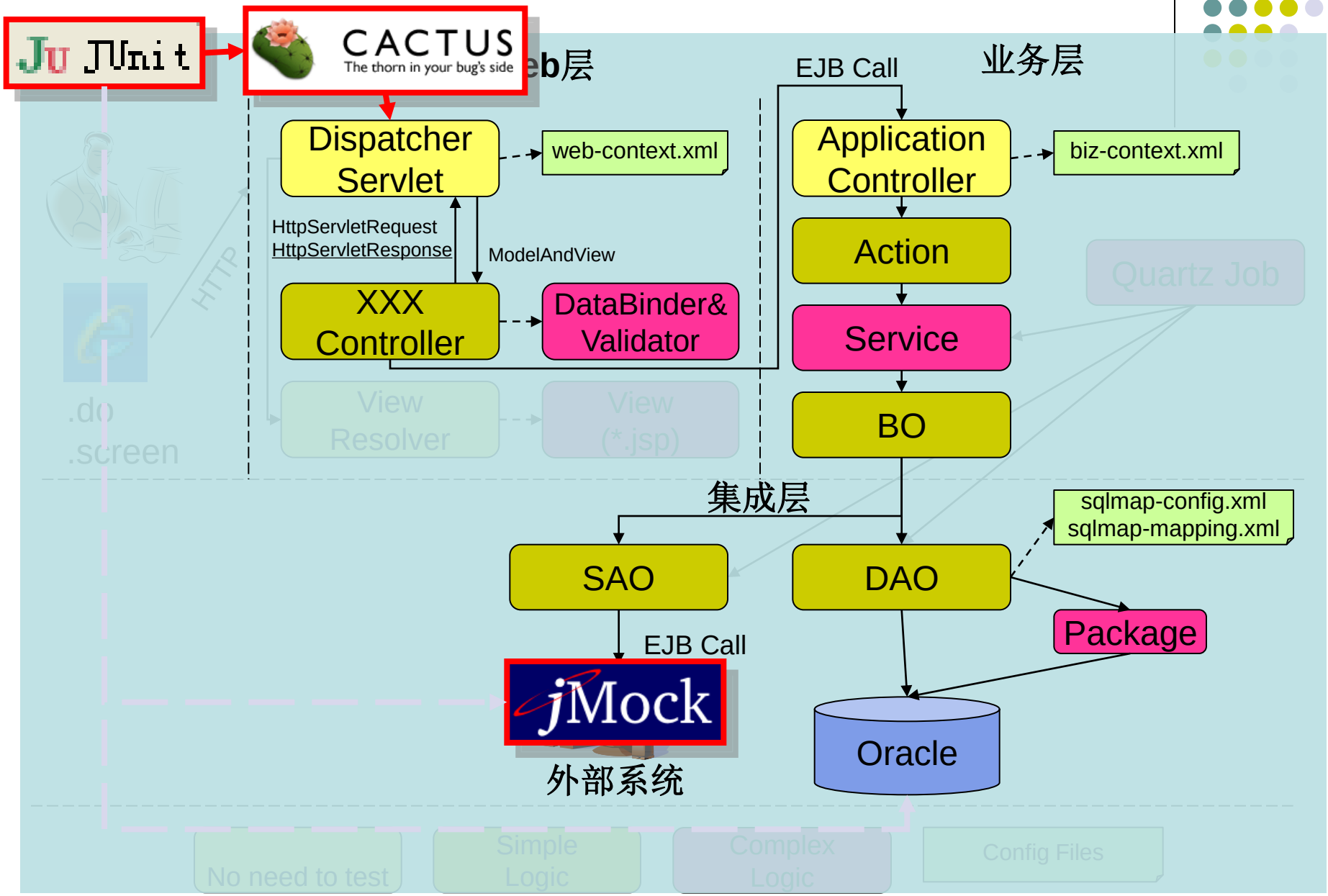
针对Package和Validator的开发者测试



分层测试策略

	开发者测试（DT）	控制驱动集成测试（CDIT）	界面驱动集成测试（IDIT）	验收测试（AT）性能测试
测试粒度	Service, Package, Job,…	子系统	业务领域	多个业务领域
关注的风险	一个模块内的逻辑功能错误	多个模块的集成问题，某些系统集成配置.	多个子系统的集成问题，某些系统集成配置，界面展示	关键业务功能的可用性，系统性能，可靠性
执行环境	开发者本机	开发集成测试环境	开发集成测试环境	Staging环境
测试类型	Out-Container	In-Container	In-Container	In-Container
数据库	N/A	测试数据	测试数据	类生产数据

控制驱动集成测试

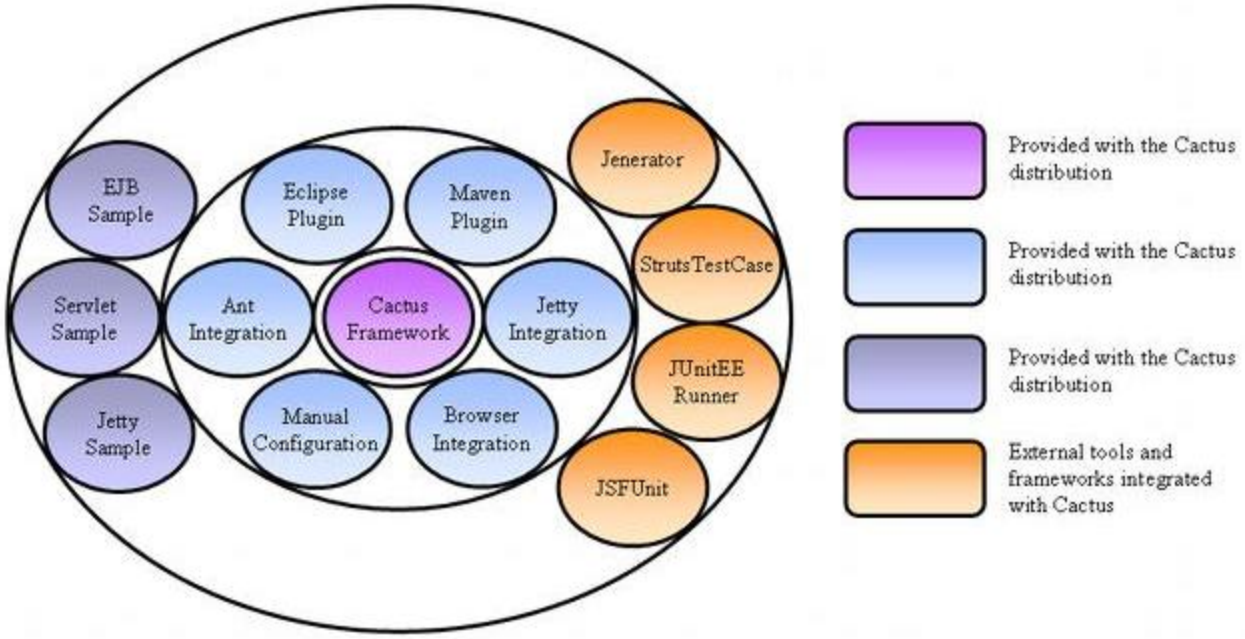

JUnit
CACTUS
The thorn in your bug's side
eb层
业务层
EJB Call
Dispatcher Servlet
web-context.xml
Application Controller
biz-context.xml
HttpServletRequest
HttpServletResponse
ModelAndView
Action
Quartz Job
XXX Controller
DataBinder& Validator
Service
HTTP
View Resolver
View (*.jsp)
BO
.do
.screen
集成层
sqlmap-config.xml
sqlmap-mapping.xml
SAO
DAO
Package
EJB Call
jMock
Oracle
外部系统
No need to test
Simple Logic
Complex Logic
Config Files

控制驱动集成测试

- 特点：
  - 绕开界面接口稳定，测试用例便于复用
- 难点：
  - 需要构造测试数据
  - 需要构造外部系统的Mock EJB

Cactus

- Cactus is a simple test framework for unit testing server-side java code (Servlets, EJBs, Tag Libs, Filters, ...).
- The intent of Cactus is to lower the cost of writing tests for server-side code. It uses JUnit and extends it.
- Cactus implements an in-container strategy, meaning that tests are executed inside the container.

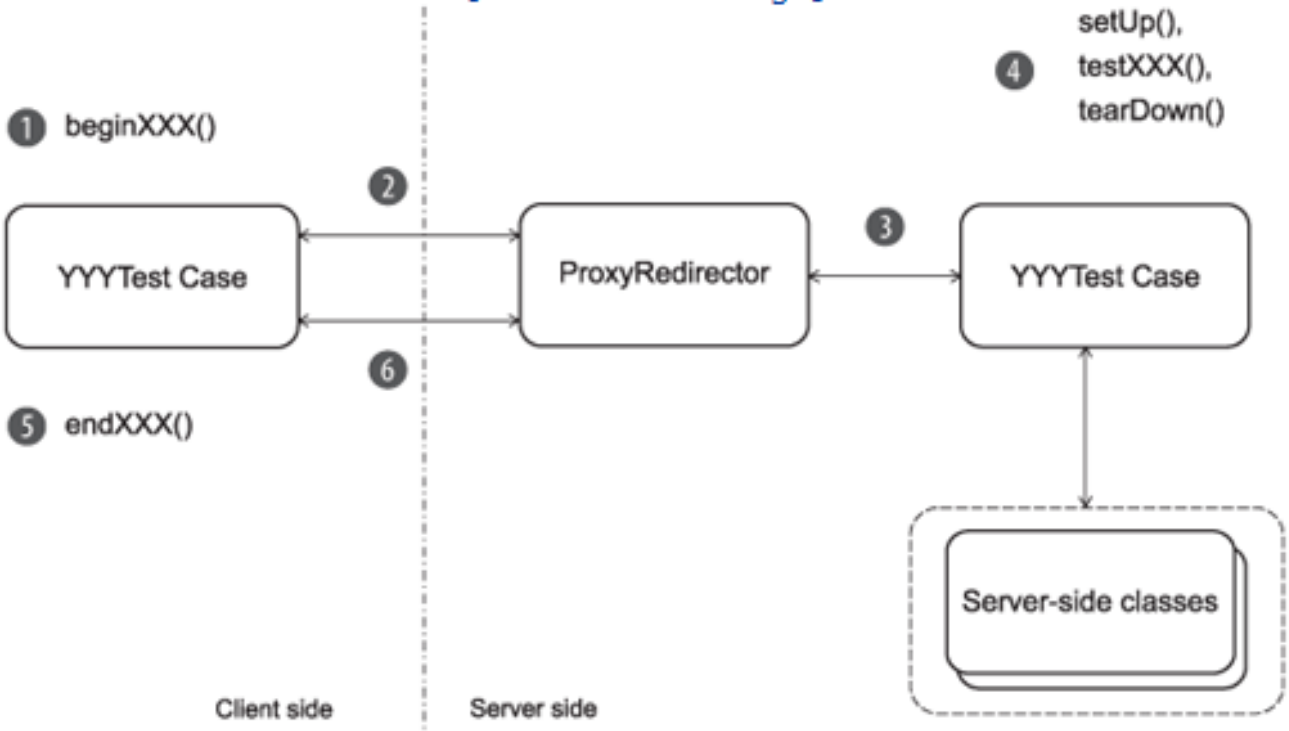


集成测试 - Cactus

- In container testing:

Figure 14.1. Lifecycle of a Cactus test

[View full size image]



集成测试 – Mock Support from Spring

```
public class HomeControllerTest  extends AbstractDependencyInjectionSpringContextTests{
    public static Test suite()
    {
        ServletTestSuite suite = new ServletTestSuite();
        suite.addTestSuite(HomeControllerTest.class);
        return suite;
    }
    // this instance will be (automatically) dependency injected
    protected DVDInfoDao dvdInfoDao;
    public HomePageController homePageController;

    ....

    protected void onSetUp() throws Exception {
        homePageController = new HomePageController();
    }

    public void testHomePageController() throws Exception{
        MockHttpServletRequest request = new MockHttpServletRequest() ;
        request.setMethod( "GET") ;
        request.setRequestURI("http://localhost:8080/mydvdweb2/home.htm" ) ;

        HttpServletResponse response = new MockHttpServletResponse();

        homePageController.setDvdInfoDao(dvdInfoDao);

        ModelAndView modelAndView =  homePageController.handleRequest(request, response);
        assertNotNull(modelAndView);
    }
}
```

集成测试 – Mock External EJB

```
public class StorageImpl extends MockStorageImplTestCase implements StorageRemote, StorageLocal  {
    private StorageLocal mockCore;
    /**
     * Default constructor.
     */
    public StorageImpl() {
        // TODO Auto-generated constructor stub
    }
    @Override
    public void createRemoteMock() {
        mockCore = mock(StorageLocal.class);
    }
    public void remoteExpects( final String expectedInput, final int expectedOut ){
        checking( new Expectations(){{
            oneOf (mockCore).getStorageCount(expectedInput);
            will(returnValue(expectedOut));
        }});
    }
    public int getStorageCount(String title){
        return mockCore.getStorageCount(title);
    }
    public void remoteVerify() throws RemoteException{
        try {
            verify();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            throw new RemoteException("Mock Verification Failed!",e.getCause());
        }
    }
}
```

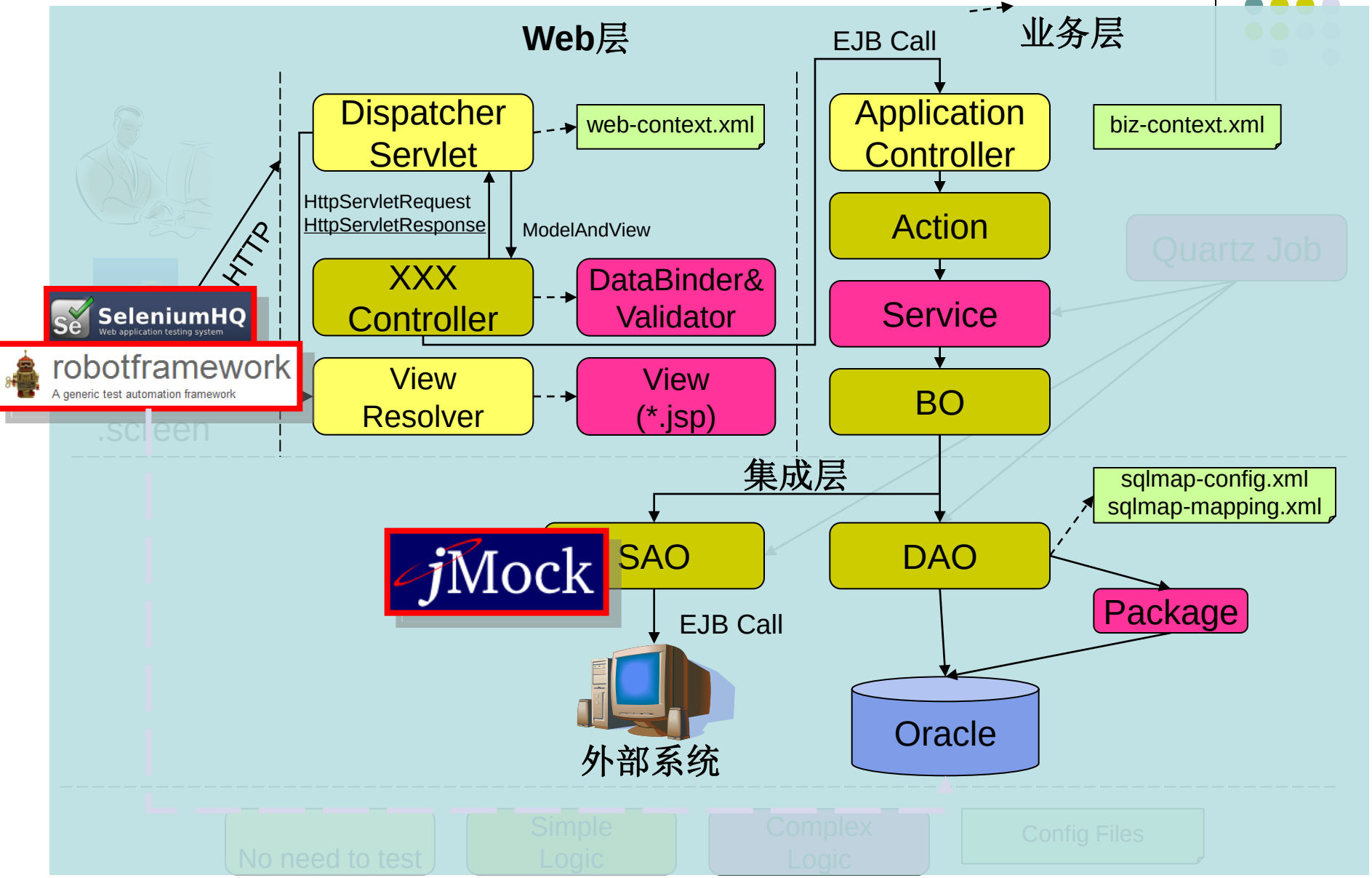
集成测试 – Mock External EJB

```
public class HomeControllerTest  extends AbstractDependencyInjectionSpringContextTests{

    ...

    public void testHomePageController() throws Exception{
        ...

        Properties props = new Properties();
        props.put(Context.INITIAL_CONTEXT_FACTORY,"org.apache.openejb.client.RemoteInitialContextFactory
        props.put(Context.PROVIDER_URL,"ejbd://127.0.0.1:4201");
        Context ctx = new InitialContext(props);
        Object ref = ctx.lookup("StorageImplRemote");
        StorageRemote s = (StorageRemote) ref;

        s.createRemoteMock();

        s.remoteExpects("",4);

        ModelAndView modelAndView =  homePageController.handleRequest(request, response);
        assertNotNull(modelAndView);

        s.remoteVerify();
    }
}
```

分层测试策略

	开发者测试（DT）	控制驱动集成测试（CDIT）	界面驱动集成测试（IDIT）	验收测试（AT）性能测试
测试粒度	Service, Package, Job,…	子系统	业务领域	多个业务领域
关注的风险	一个模块内的逻辑功能错误	多个模块的集成问题，某些系统集成配置.	多个子系统的集成问题，某些系统集成配置，界面展示	关键业务功能的可用性，系统性能，可靠性
执行环境	开发者本机	开发集成测试环境	开发集成测试环境	Staging环境
测试类型	Out-Container	In-Container	In-Container	In-Container
数据库	N/A	测试数据	测试数据	类生产数据

界面驱动集成测试


Web层
业务层
EJB Call
Dispatcher Servlet
web-context.xml
Application Controller
biz-context.xml
HttpServletRequest
HttpServletResponse
ModelAndView
Action
Quartz Job
HTTP
XXX Controller
DataBinder& Validator
Service
SeleniumHQ
Web application testing system
robotframework
A generic test automation framework
View Resolver
View (*.jsp)
BO
.screen
集成层
sqlmap-config.xml
sqlmap-mapping.xml
jMock
SAO
DAO
Package
EJB Call
外部系统
Oracle
No need to test
Simple Logic
Complex Logic
Config Files

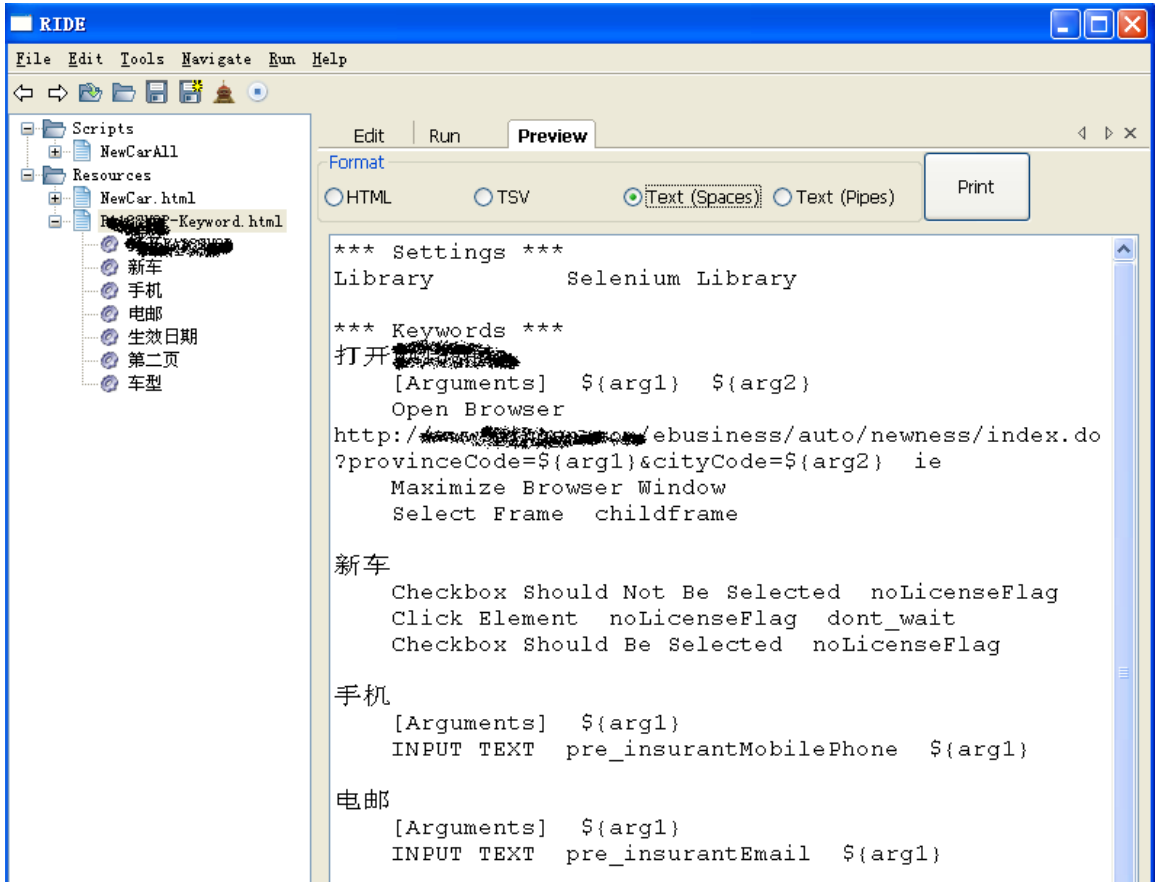
界面稳定性测试

- Java + Selenium 2（WebDriver)
- 主要目标是验证界面元素是否有变化

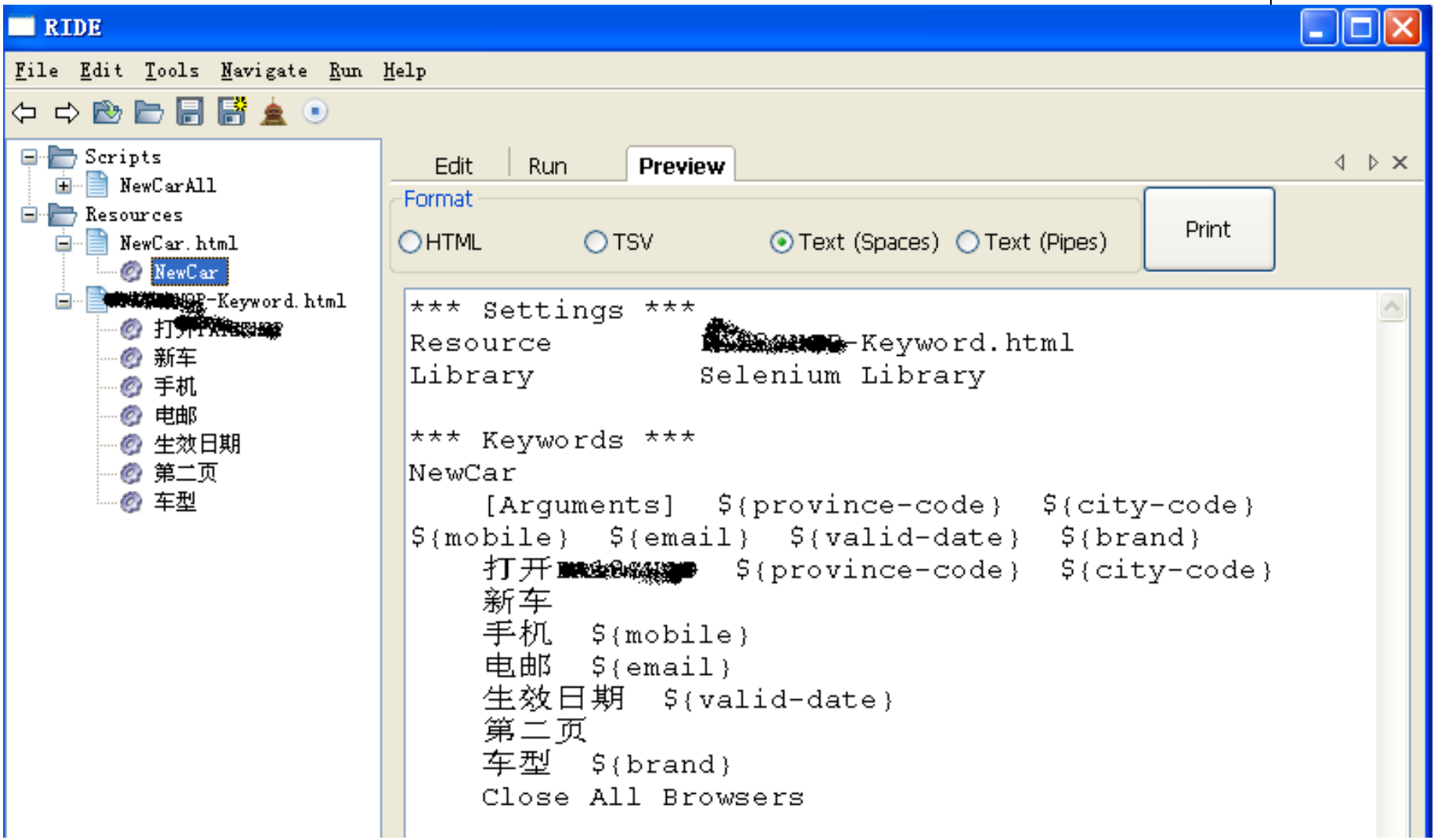
```
public class WebDriverTest {
    public static void main(String[] args) throws Exception {
        // The Firefox driver supports javascript
//      WebDriver driver = new HtmlUnitDriver();
        WebDriver driver = new FirefoxDriver();

        // Go to the Google Suggest home page
        driver.get("http://chexian.xxx.com/");

        // Enter the query string "北京"
        WebElement city = driver.findElement(By.id("select_city"));
        do{
            Thread.sleep(1000);
            System.out.println(Boolean.toString((city.isEnabled())));
        }while(city.isEnabled()==false);
        city.click();

        WebElement list = driver.findElement(By.className("list-f"));
        list.click();

        WebElement button = driver.findElement(By.className("a_lq_btn03"));
        button.click();
```

关键词驱动测试

- 利用Robot Framework的关键词封装界面实现细节



基于关键词构建测试脚本模版



数据驱动测试

RIDE

File Edit Tools Navigate Run Help

- Scripts
  - NewCarAll
    - BeijingNewCar
- Resources
  - NewCar.html
    - NewCar
  - PA18SHOP-Keyword.html
    - 打开PA18SHOP
    - 新车
    - 手机
    - 电邮
    - 生效日期
    - 第二页
    - 车型

Edit | Run | **Preview**

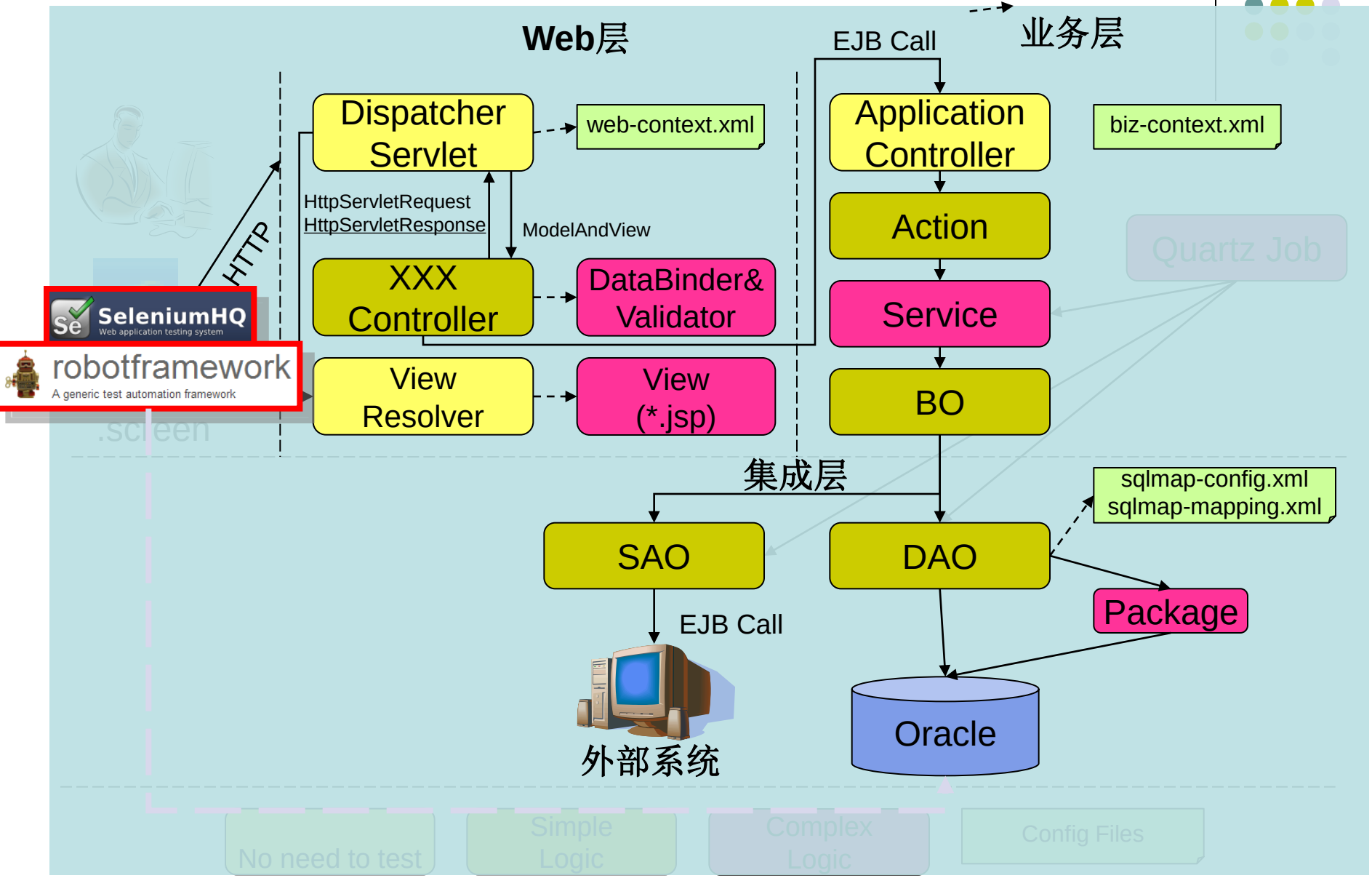
Format: HTML | TSV | Text (Spaces) | Text (Pipes) | Print

```
*** Settings ***
Suite Setup      Start Selenium Server
Suite Teardown   Stop Selenium Server
Test Template    NewCar
Test Timeout     30 seconds
Library          Selenium Library
Resource         NewCar.html
Resource         PA18SHOP-Keyword.html

*** Test Cases ***
BeijingNewCar
    [Template]
    NewCar  110000  110100  13333333333
adwu73@hotmail.com  2011-4-10  本田
    [Teardown]
```

界面驱动集成测试

- 特点
  - 界面变化快，测试用例维护工作量大，因此，测试用例数量必须严格控制
- 难点
  - 需要构造多个子系统的测试数据

验收测试


Web层
业务层
EJB Call
Dispatcher Servlet
web-context.xml
Application Controller
biz-context.xml
HttpServletRequest
HttpServletResponse
ModelAndView
Action
Quartz Job
HTTP
XXX Controller
DataBinder& Validator
Service
SeleniumHQ
Web application testing system
robotframework
A generic test automation framework
View Resolver
View (*.jsp)
BO
.screen
集成层
sqlmap-config.xml
sqlmap-mapping.xml
SAO
DAO
Package
EJB Call
外部系统
Oracle
No need to test
Simple Logic
Complex Logic
Config Files

验收测试

- 特点
  - 可以重用界面驱动集成测试的测试用例
- 难点
  - 需要一套稳定的类生产数据用于测试

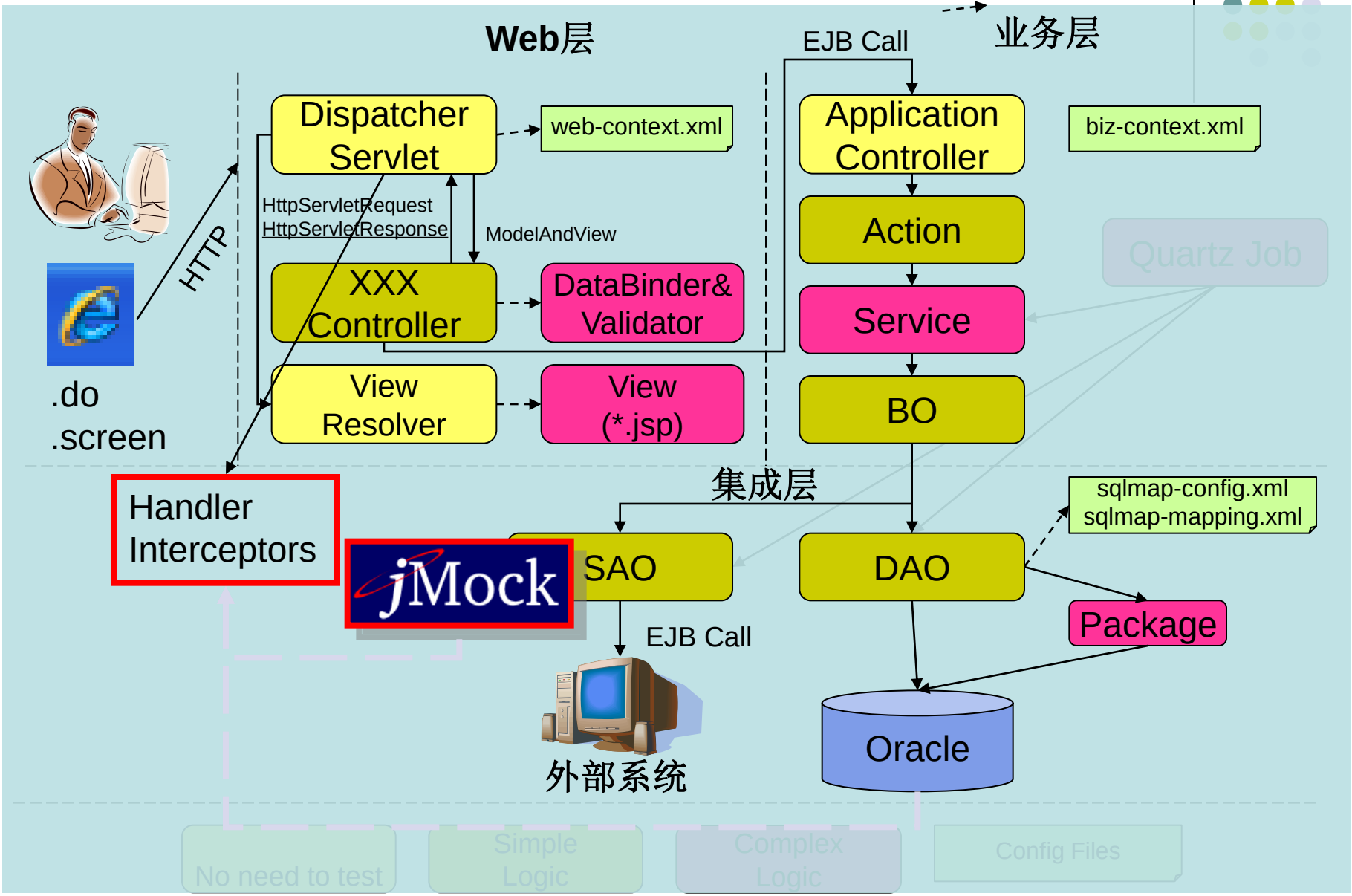
融合测试方式

- 开发测试
  - 开发人员负责编写测试用例
  - 测试人员复审测试用例
- 控制驱动集成测试
  - 开发人员提供框架
  - 测试人员提供测试数据
- 界面驱动集成测试，验收测试
  - 测试自动化人员提供框架
  - 测试人员提供测试数据

自动化测试用例生成

- 可以考虑通过录制方式自动生成单元测试以及控制驱动集成测试的测试用例，降低开发人员的工作量；
  - 利用Spring中的HandlerInterceptors

自动化测试用例生成


Web层
EJB Call
业务层
Dispatcher Servlet
web-context.xml
Application Controller
biz-context.xml
HttpServletRequest
HttpServletResponse
ModelAndView
Action
Quartz Job
HTTP
XXX Controller
DataBinder& Validator
Service
View Resolver
View (*.jsp)
BO
.do
.screen
集成层
Handler Interceptors
sqlmap-config.xml
sqlmap-mapping.xml
jMock
SAO
DAO
Package
EJB Call
外部系统
Oracle
No need to test
Simple Logic
Complex Logic
Config Files

Backup Slides

QCon
杭州站 · 2011年10月20日~22日
www.qconhangzhou.com（6月启动）
QCon北京站官方网站和资料下载
www.qconbeijing.com