JUnit单元测试

1. 单元测试的目的

提高代码质量，快速发布，快速迭代

1. 哪些代码要加单元测试
   1. 业务机能的代码，只写Action的测试即可，不必写业务service的测试
   2. 工具类的方法，一定要写工具类的测试
   3. 如果只提供service方法，写service的测试
   4. 自定义的Dao方法，例如CollectionManageDao的findMaxRegisterNum方法
2. 命名
   1. 测试类：【被测试类名】+Test
   2. 测试方法：test【被测试方法】

同一个方法有多个测试方法，test【被测试方法】1, test【被测试方法】2

* 1. 测试类所在的包：和测试类相同的包结构，放到src/test/java文件夹下。
  2. 测试类使用到的配置文件：（相关xml文件命名，以action的测试为例）
  3. 文件位置：

/src/test/resources/com/ld/collections/action/register

注意在action包下，建一级新的包，跟dto所在的包的命名一样即可

* 1. 准备数据使用的xml，一般来说，一个测试方法对应一个

[要测试的方法名]\_db\_setup[xxx].xml

例如：

/LongMuseumU/src/test/resources/com/ld/collections/action/register/init\_db\_setup.xml

如果一个方法写了多个测试方法，用序号分开。

* 1. 存放期望数据的xml，类似准备数据的xml

[要测试的方法名]\_db\_expected[xxx].xml

例如：

/LongMuseumU/src/test/resources/com/ld/collections/action/register/saveBasic\_db\_expected1.xml

如果一个方法写了多个测试方法，用序号分开。

* 1. 注入要比较的bean使用的xml，一般来说，一个测试类对应一个，

并且要注意\*\*\*\_bean.xml所在的文件夹中所有的bean的id要唯一

A.文件命名：[大机能]\_bean[n].xml

例如: register\_bean.xml

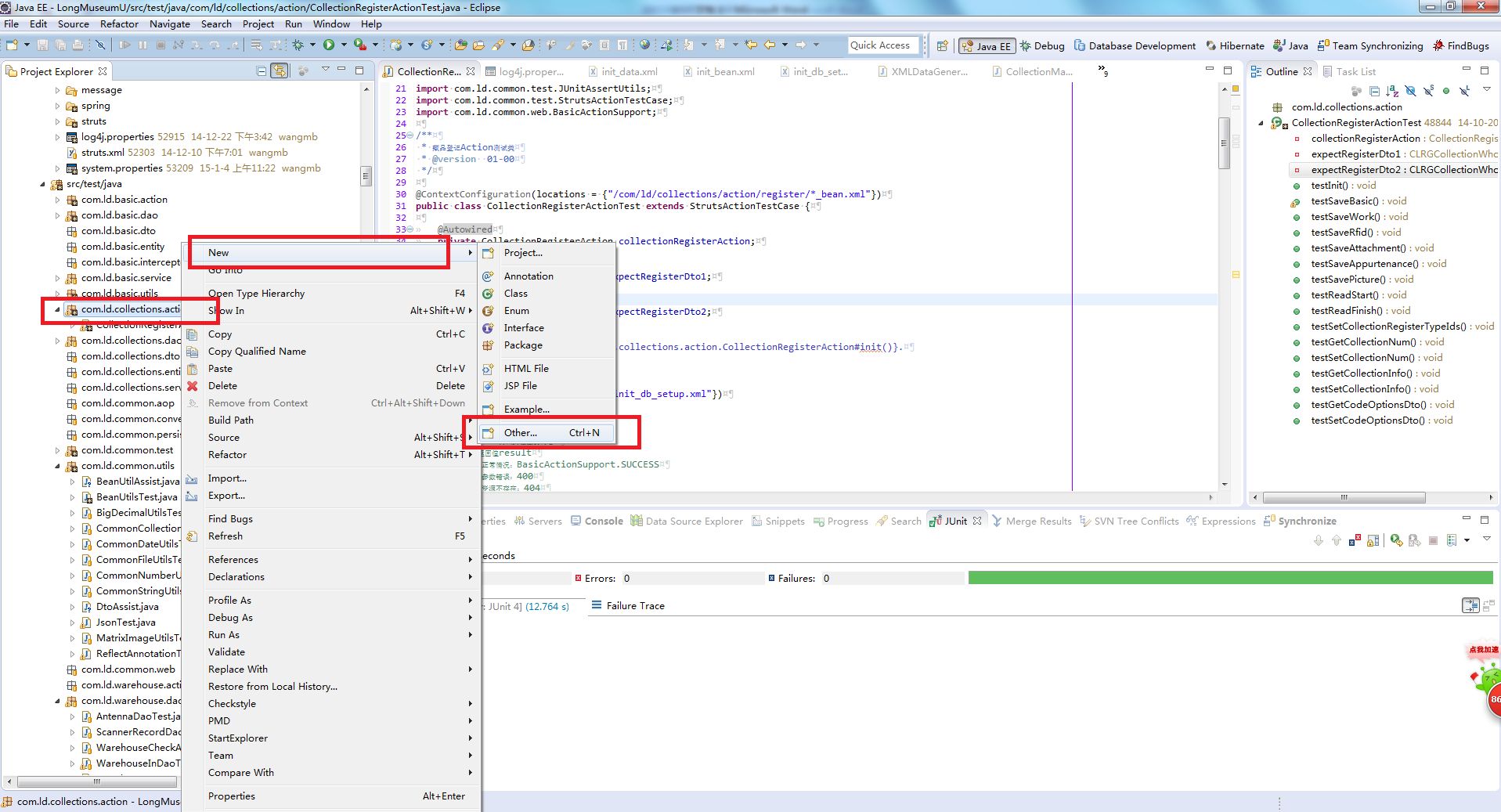
如果有多个，后面加数字

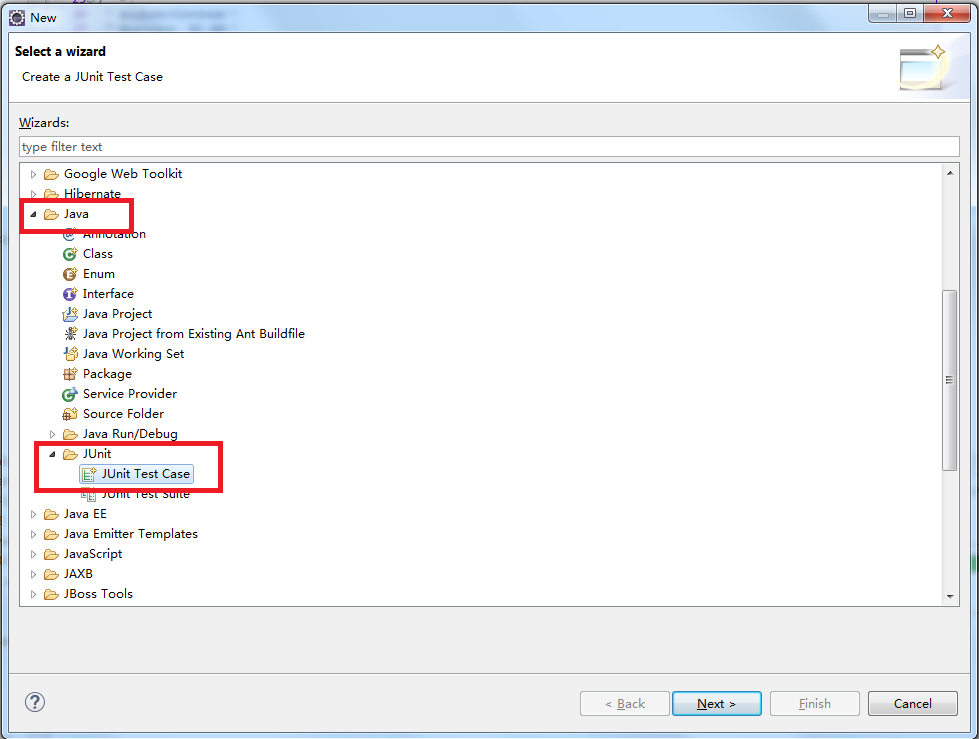
B.bean的id的命名：[方法名]XXXX[n]

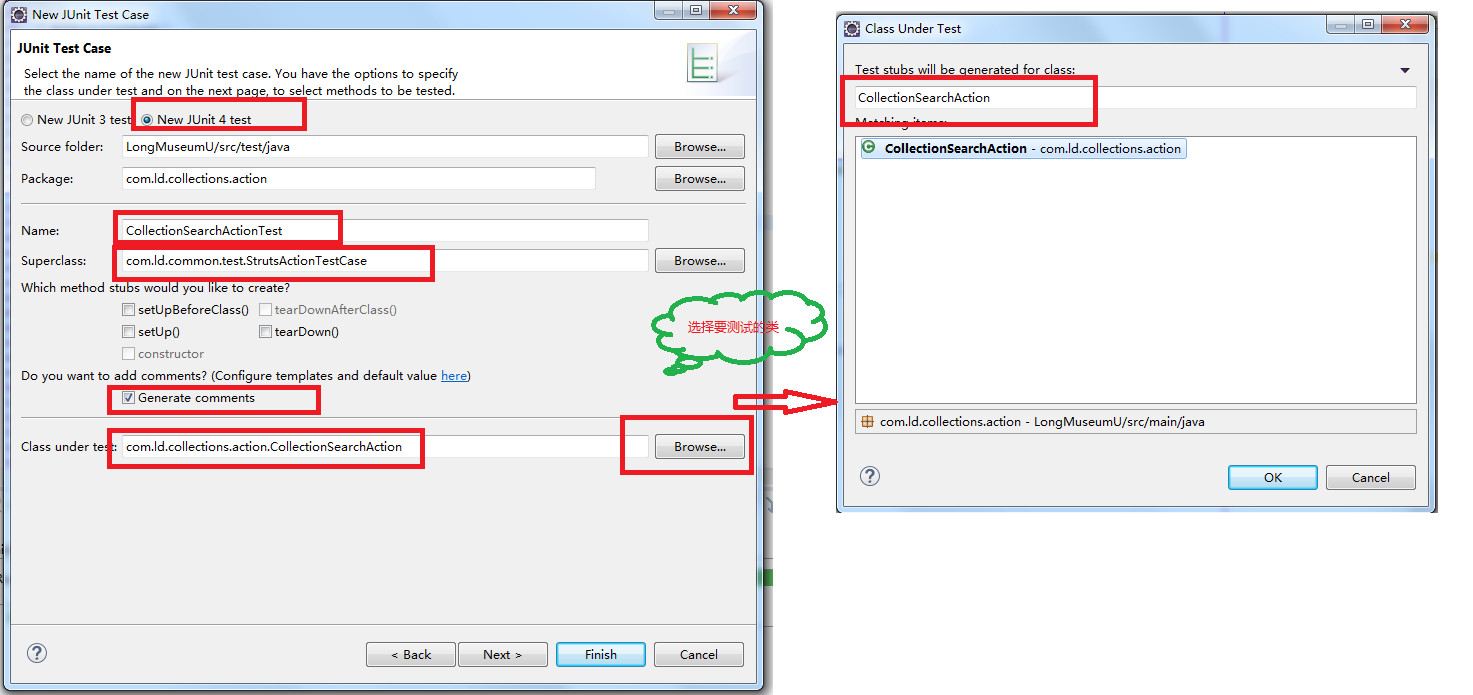
例如：

|  |
| --- |
| <bean id="initBasicInfo1" class="com.ld.collections.dto.register.CLRGCollectionBasicDto">  </bean>  <bean id="initWorkInfo1" class="com.ld.collections.dto.register.CLRGCollectionWorkInfoDto">  </bean>  <bean id="initRegisterDto1" class="com.ld.collections.dto.register.CLRGCollectionWholeDto">  <property name="basicInfo" ref="initBasicInfo1"/>  <property name="workInfo" ref="initWorkInfo1"/>  </bean>  <bean id="initRegisterDto2" class="com.ld.collections.dto.register.CLRGCollectionWholeDto">  <property name="basicInfo">  <bean class="com.ld.collections.dto.register.CLRGCollectionBasicDto">  </bean>  </property>  <property name="workInfo">  <bean class="com.ld.collections.dto.register.CLRGCollectionWorkInfoDto">  </bean>  </property>  </bean> |

1. 使用Eclipse生成JUnit测试类（以Action的测试为例）





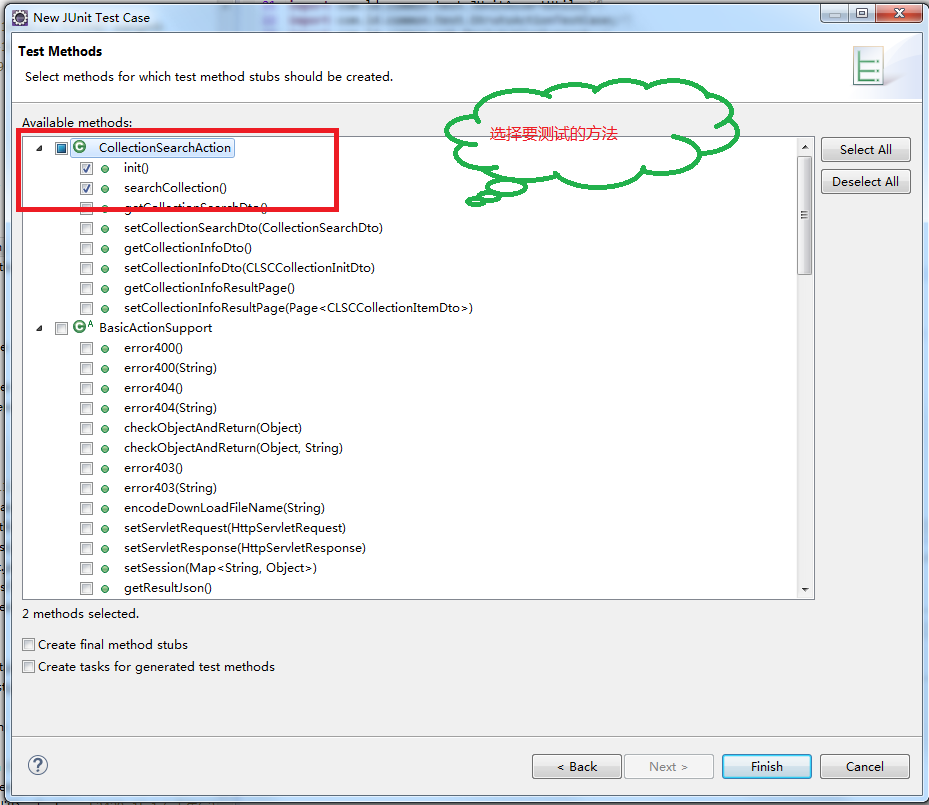


一般的：  
Action的测试类继承自

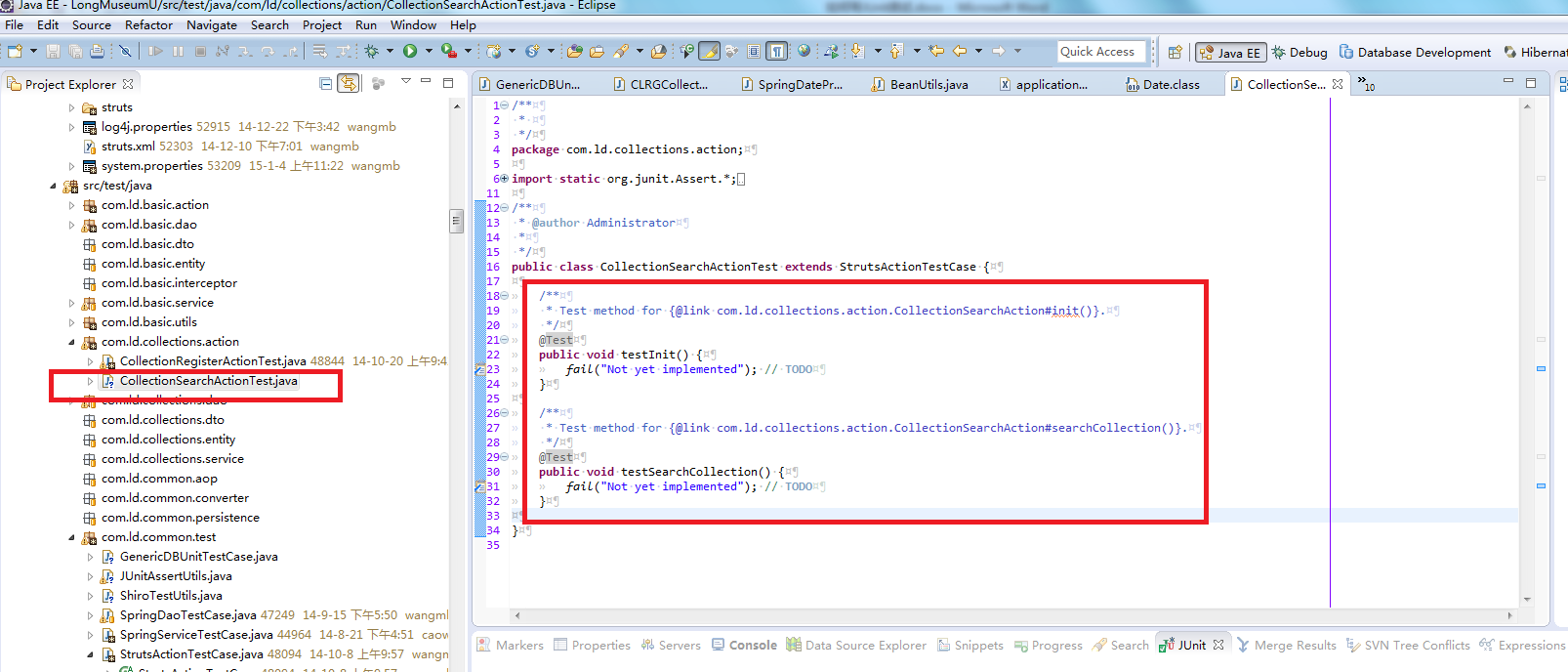
com.ld.common.test.StrutsActionTestCase

其他的需要访问数据库的测试类，例如Dao和Service的测试类，继承自com.ld.common.test.GenericDBUnitTestCase

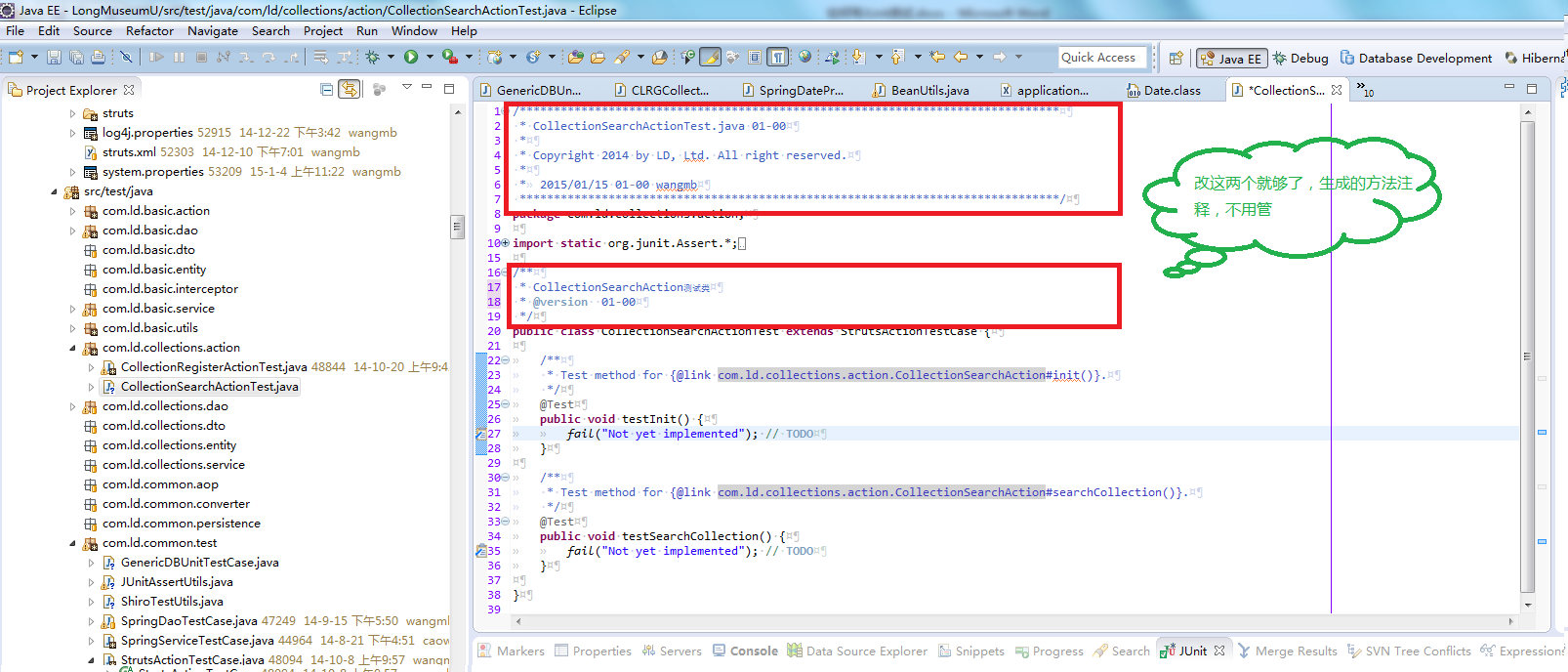
其他的测试类，例如工具类，不需要集成其他的类



Eclipse帮我们生成了测试类和方法：



修改注释：



注入：被测试的Action类：

|  |
| --- |
| **public** **class** CollectionRegisterActionTest **extends** StrutsActionTestCase {  @Autowired  **private** CollectionRegisterAction collectionRegisterAction; |

1. 测试用例设计：

1.正常：

2.异常：

2.1.传入参数错误

2.2.资源不存在

3.其他满足覆盖率的用例设计

1. Action测试要检查的内容和方法
2. 返回值result

正常情况：BasicActionSupport.SUCCESS

参数错误：400

资源不存在：404

1. struts2 error检查

正常情况应该没有

|  |
| --- |
| JUnitAssertUtils.*noStrutsErrors*(collectionRegisterAction); |

如果有错误，检查是否是期望的错误

collectionRegisterAction.getErrorMessages();

collectionRegisterAction.getActionErrors();

collectionRegisterAction.getFieldErrors();

collectionRegisterAction.getErrors();

1. 数据库操作是否正确
   1. 取数据，取到的是不是正确的数据

方法：

|  |
| --- |
| /\*\*  \* Case 1. 正常case。  \*/  collectionNum = "A0000001";  collectionRegisterAction.setCollectionNum(collectionNum);  result = collectionRegisterAction.init();  registerDto = collectionRegisterAction.getRegisterDto();  Assert.*assertEquals*(BasicActionSupport.***SUCCESS***, result);  JUnitAssertUtils.*noStrutsErrors*(collectionRegisterAction);  JUnitAssertUtils.*equalGivenProperties*(initRegisterDto1, registerDto, **new** String[] {"basicInfo", "workInfo"}); |

说明：

|  |
| --- |
| registerDto是被测试方法返回的对象  initRegisterDto1是用于作比较的对象，即期望registerDto是和initRegisterDto1相等的对象  **new** String[] {"basicInfo", "workInfo"}是要检查的DTO中的属性 |

工具方法的使用：

A.相等判断

|  |
| --- |
| JUnitAssertUtils.equalGivenProperties  JUnitAssertUtils.equal |

B.使用spring注入期望的对象，不需要写代码构造initRegisterDto1对象

|  |
| --- |
| @ContextConfiguration(locations = {"/com/ld/collections/action/register/\*\_bean.xml"})  **public** **class** CollectionRegisterActionTest **extends** StrutsActionTestCase {  @Autowired  **private** CollectionRegisterAction collectionRegisterAction;  @Autowired  **private** CLRGCollectionWholeDto initRegisterDto1; |

register\_bean.xml文件

|  |
| --- |
| <bean id=*"initBasicInfo1"* class=*"com.ld.collections.dto.register.CLRGCollectionBasicDto"*>  <property name=*"collectionNum"* value=*"A0000001"*/>  <property name=*"oldCollectionName"* value=*"寿桃"*/>  <property name=*"collectionName"* value=*"刘小东《寿桃》布面油彩 200×200cm 2010"*/>  <property name=*"basicType"* value=*"Л2801Л2802Л2803Л2804Л2805Л"*/>  <property name=*"historicalTypeLarge"* value=*"0101"*/>  <property name=*"historicalTypeSmall"* value=*"Л02Л"*/>  <property name=*"sampleType"* value=*"Л0201Л0202Л0203Л0204Л0205Л"*/>  <property name=*"modelType"* value=*"Л0301Л0302Л0303Л"*/>  <property name=*"copyType"* value=*"Л0401Л0402Л"*/>  <property name=*"periodType"* value=*"Л0501Л0502Л0503Л0504Л"*/>  <property name=*"period"* value=*"Л2010Л2011Л2014Л2016Л"*/>  <property name=*"areaType"* value=*"Л0601Л0602Л0603Л0604Л"*/>  <property name=*"area"* value=*"Л1域Л2域Л3域Л4域Л"*/>  <property name=*"culturalType"* value=*"Л0701Л0702Л0703Л0704Л"*/>  <property name=*"culturalBuilderId"* value=*"AU000001"*/>  <property name=*"culturalBuilder"*>  <bean class=*"com.ld.basic.entity.ArtistInfo"*>  <property name=*"artistId"* value=*"AU000001"*/>  <property name=*"artistName"* value=*"刘小东"*/>  <property name=*"artistAlias"* value=*""*/>  <property name=*"artistZi"* value=*""*/>  <property name=*"artistHao"* value=*""*/>  <property name=*"birthplace"* value=*""*/>  <property name=*"deadPlace"* value=*""*/>  <property name=*"birthday"* value=*""*/>  <property name=*"deadDay"* value=*""*/>  <property name=*"gender"* value=*""*/>  <property name=*"nationality"* value=*""*/>  <property name=*"period"* value=*""*/>  <property name=*"picture1"* value=*""*/>  <property name=*"picture2"* value=*""*/>  <property name=*"artistDesc"* value=*""*/>  <property name=*"contact"* value=*""*/>  <property name=*"contactRelation"* value=*""*/>  <property name=*"contactAddress"* value=*""*/>  <property name=*"contactPostcode"* value=*""*/>  <property name=*"contactTelephone"* value=*""*/>  <property name=*"contactMobile"* value=*""*/>  <property name=*"contactMemo"* value=*""*/>  <property name=*"createDateTime"* value=*"2014-09-17 12:00:00.0"*/>  <property name=*"createUserId"* value=*"U9999999"*/>  <property name=*"updateDateTime"* value=*"2014-09-17 12:00:00.0"*/>  <property name=*"updateUserId"* value=*"U9999999"*/>  <property name=*"delFlag"* value=*"false"*/>  </bean>  </property>  <property name=*"culturalCopyerId"* value=*"AU000030"*/>  <property name=*"culturalCopyer"*>  。。。。。。  </property>  <property name=*"culturalUserId"* value=*"AU000070"*/>  <property name=*"culturalUser"*>  。。。。。。  </property>  <property name=*"culturalOwnerId"* value=*"AU000091"*/>  <property name=*"culturalOwner"*>  。。。。。。  </property>  <property name=*"textureTypeLarge"* value=*"0801"*/>  <property name=*"textureTypeSmall"* value=*"Л01Л02Л03Л04Л"*/>  <property name=*"textureTypeChild"* value=*"布面油彩"*/>  <property name=*"featureTypeLarge"* value=*"0901"*/>  <property name=*"featureTypeSmall"* value=*"Л01Л02Л03Л04Л05Л06Л07Л"*/>  <property name=*"actualUsage"* value=*"A0711 实 际 用 途"*/>  <property name=*"polytechnic"* value=*"Л1001Л1002Л1015Л"*/>  <property name=*"sect"* value=*"A0811 流 派"*/>  <property name=*"drawMethod"* value=*"A0812 绘 画 技 法"*/>  <property name=*"decoratePolytechnic"* value=*"A0821 装 饰 生 成 工 艺"*/>  <property name=*"literalPolytechnic"* value=*"A0822 文字生成工艺"*/>  <property name=*"objectType"* value=*"Л1101Л1102Л1104Л"*/>  <property name=*"shape"* value=*"A0911 形 态 特 征"*/>  <property name=*"uniqueFeature"* value=*"A0912 独 特 标 志"*/>  <property name=*"completeTypeLarge"* value=*"1201"*/>  <property name=*"completeTypeSmall"* value=*"01"*/>  <property name=*"complete"* value=*"完 残 状 况"*/>  <property name=*"colorType"* value=*"1301"*/>  <property name=*"color"* value=*"A1110 颜色"*/>  <property name=*"lusterType"* value=*"1401"*/>  <property name=*"luster"* value=*"A1120 光泽"*/>  <property name=*"decorateStyleType"* value=*"1501"*/>  <property name=*"decorateSubjectType"* value=*"A1202 装 饰 题 材 类 别"*/>  <property name=*"combinationTypeLarge"* value=*"1601"*/>  <property name=*"combinationTypeSmall"* value=*"Л02Л03Л"*/>  <property name=*"decoratePosition"* value=*"A1204 施 饰 部 位"*/>  <property name=*"posy"* value=*"A1301 铭 文"*/>  <property name=*"kuanShi"* value=*"A1302 款 识"*/>  <property name=*"tiBa"* value=*"A1303 题 跋"*/>  <property name=*"autograph"* value=*"A1304 题 名"*/>  <property name=*"impression"* value=*"A1305 印 鉴"*/>  <property name=*"summary"* value=*"A1306 内容提要"*/>  <property name=*"literalTypeLarge"* value=*"1703"*/>  <property name=*"literalTypeSmall"* value=*"Л99Л"*/>  <property name=*"literalTypeOther"* value=*"A1401 文 字 种 类(其 他)"*/>  <property name=*"fontTypeLarge"* value=*"1801"*/>  <property name=*"fontTypeSmall"* value=*"Л03Л04Л"*/>  <property name=*"fontColor"* value=*"A1403 字 迹 颜 色"*/>  <property name=*"font"* value=*"A1410 字 体"*/>  <property name=*"fossilType"* value=*"Л1901Л1902Л1903Л"*/>  <property name=*"fossil"* value=*"Л植物Л动物Л古人类Л"*/>  <property name=*"patternType"* value=*"A1502 模 式 类 别"*/>  <property name=*"growthProgress"* value=*"A1503 发育阶段"*/>  <property name=*"gerder"* value=*"A1504 性别"*/>  <property name=*"size"* value=*"200×200"*/>  <property name=*"sizeUnit"* value=*"2002"*/>  <property name=*"capacity"* value=*"1621"*/>  <property name=*"capacityUnit"* value=*"2101"*/>  <property name=*"quality"* value=*"1631"*/>  <property name=*"qualityUnit"* value=*"2202"*/>  <property name=*"actualCount"* value=*"2"*/>  <property name=*"actualUnit"* value=*"2302"*/>  <property name=*"traditionCount"* value=*"1"*/>  <property name=*"traditionUnit"* value=*"2401"*/>  <property name=*"memo"* value=*"A2002 备 注"*/>  <property name=*"galleryType"* value=*"2909"*/>  <property name=*"areaCode"* value=*"3"*/>  <property name=*"statustFlag"* value=*"3511"*/>  <property name=*"currentPosition"* value=*"浦东库房"*/>  </bean>  <bean id=*"initWorkInfo1"* class=*"com.ld.collections.dto.register.CLRGCollectionWorkInfoDto"*>  <property name=*"collectionNum"* value=*"A0000001"*/>  <property name=*"origin"* value=*"2602"*/>  <property name=*"originSide"* value=*"和辉艺术"*/>  <property name=*"originNum"* value=*"B0103 来 源 号"*/>  <property name=*"discoveryDateTime"* value=*"出土时间1952"*/>  <property name=*"discoveryPlace"* value=*"出土地点"*/>  <property name=*"discoveryPerson"* value=*"发掘人"*/>  <property name=*"discoveryProcess"* value=*"B0104 搜集经过"*/>  <property name=*"applyFund"* value=*"111.00"*/>  <property name=*"actualFund"* value=*"112.00"*/>  <property name=*"spreadProcess"* value=*"B0121 流传经历"*/>  <property name=*"collectionOrganization"* value=*"B0201 收藏单位"*/>  <property name=*"inLibNum"* value=*"B0211 入 馆 登 记 号"*/>  <property name=*"inLibDate"* value=*"2013-01-14"*/>  <property name=*"registerNum"* value=*"1"*/>  <property name=*"classifyNum"* value=*"C-1"*/>  <property name=*"delFlag"* value=*"false"*/>  </bean>  <bean id=*"initRegisterDto1"* class=*"com.ld.collections.dto.register.CLRGCollectionWholeDto"*>  <property name=*"basicInfo"* ref=*"initBasicInfo1"*/>  <property name=*"workInfo"* ref=*"initWorkInfo1"*/>  </bean>  <bean id=*"initRegisterDto2"* class=*"com.ld.collections.dto.register.CLRGCollectionWholeDto"*>  <property name=*"basicInfo"*>  <bean class=*"com.ld.collections.dto.register.CLRGCollectionBasicDto"*>  </bean>  </property>  <property name=*"workInfo"*>  <bean class=*"com.ld.collections.dto.register.CLRGCollectionWorkInfoDto"*>  </bean>  </property>  </bean> |

* 1. 新增数据，结果是否正确

|  |
| --- |
| /\*\*  \* Test method for {@link com.ld.collections.action.CollectionRegisterAction#saveBasic()}.  \* Case 1.1. 没有传入dto对象时  \* Case 1.2. 新增操作  \*/  @Test  @DatabaseSetup(value={"register/saveBasic\_db\_setup1.xml"}, type=DatabaseOperation.***DELETE\_ALL***)  @ExpectedDatabase(value=("register/saveBasic\_db\_expected1.xml"), assertionMode=DatabaseAssertionMode.***NON\_STRICT***)  @DatabaseTearDown(value={"register/saveBasic\_db\_setup1.xml"}, type=DatabaseOperation.***DELETE\_ALL***)  **public** **void** testSaveBasic1() { |

简单说明：

方法执行开始要先清理数据（要将此次要操作的数据库表中的数据清理掉）

@DatabaseSetup(value={"register/saveBasic\_db\_setup1.xml"}, type=DatabaseOperation.***DELETE\_ALL***)

方法执行开始要先清理数据

@DatabaseTearDown(value={"register/saveBasic\_db\_setup1.xml"}, type=DatabaseOperation.***DELETE\_ALL***)

比较数据的方法

@ExpectedDatabase(value=("register/saveBasic\_db\_expected1.xml"), assertionMode=DatabaseAssertionMode.***NON\_STRICT***)

register/saveBasic\_db\_expected1.xml文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <dataset>  <t001 A100001=*"*A0000001*"* A100002=*"寿桃"* A100003=*"刘小东《寿桃》布面油彩 200×200cm 2010"*  A100004=*"Л2801Л2802Л2803Л2804Л2805Л"* A100005=*"0101"*  A100006=*"Л02Л"* A100007=*"Л0201Л0202Л0203Л0204Л0205Л"* A100008=*"Л0301Л0302Л0303Л"*  A100009=*"Л0401Л0402Л"* A100010=*"Л0501Л0502Л0503Л0504Л"* A100011=*"Л2010Л2011Л2014Л2016Л"*  A100012=*"Л0601Л0602Л0603Л0604Л"* A100013=*"Л1域Л2域Л3域Л4域Л"* A100014=*"Л0701Л0702Л0703Л0704Л"*  A100015=*"AU000001"* A100016=*"AU000030"* A100017=*"AU000070"* A100018=*"AU000091"*  A100019=*"0801"* A100020=*"Л01Л02Л03Л04Л"* A100021=*"布面油彩"* A100022=*"0901"*  A100023=*"Л01Л02Л03Л04Л05Л06Л07Л"* A100024=*"A0711 实 际 用 途"*  A100025=*"Л1001Л1002Л1015Л"* A100026=*"A0811 流 派"* A100027=*"A0812 绘 画 技 法"*  A100028=*"A0821 装 饰 生 成 工 艺"* A100029=*"A0822 文字生成工艺"* A100030=*"Л1101Л1102Л1104Л"*  A100031=*"A0911 形 态 特 征"* A100032=*"A0912 独 特 标 志"* A100033=*"1201"*  A100034=*"01"* A100035=*"完 残 状 况"* A100036=*"1301"* A100037=*"A1110 颜色"*  A100038=*"1401"* A100039=*"A1120 光泽"* A100040=*"1501"* A100041=*"A1202 装 饰 题 材 类 别"*  A100042=*"1601"* A100043=*"Л02Л03Л"* A100044=*"A1204 施 饰 部 位"*  A100045=*"A1301 铭 文"* A100046=*"A1302 款 识"* A100047=*"A1303 题 跋"*  A100048=*"A1304 题 名"* A100049=*"A1305 印 鉴"* A100050=*"A1306 内容提要"*  A100051=*"1703"* A100052=*"Л99Л"* A100053=*"A1401 文 字 种 类(其 他)"* A100054=*"1801"*  A100055=*"Л03Л04Л"* A100056=*"A1403 字 迹 颜 色"* A100057=*"A1410 字 体"*  A100058=*"Л1901Л1902Л1903Л"* A100059=*"A1502 模 式 类 别"* A100060=*"A1503 发育阶段"*  A100061=*"A1504 性别"* A100062=*"200×200"* A100063=*"2002"* A100064=*"1621"*  A100065=*"2101"* A100066=*"1631"* A100067=*"2202"* A100068=*"2"* A100069=*"2302"*  A100070=*"1"* A100071=*"2401"* A100072=*"A2002 备 注"* A100077=*"2909"*  S100003=*"3"* S100009=*"3511"* S100011=*"浦东库房"* C999995=*"false"*  C999992=*"U0000000"*  C999994=*"U0000000"* A100078=*"Л植物Л动物Л古人类Л"* />  <t003 A100001=*"*A0000001*"* A170014=*"1"* A170015=*"C-1"* C999995=*"false"*  C999992=*"U0000000"*  C999994=*"U0000000"* />  </dataset> |

注意：

* 以上xml中只放期望比较的字段
* 每次运行的值有可能不一样的字段不要参与判断

例如自增字段：部门id

例如在藏品登记的创建时间字段C999991，更新时间字段C999993，不要添加到上述xml中。

* 要检查创建者ID字段C999992，更新者ID字段C999994，

现在模拟的用户的ID是U0000000

* 1. 更新数据，类似追加数据，只不过更新数据的判断

主键字段和创建时间字段C999991要参与判断

并且注意，造数据时C999992和C999994要不一样

* 1. 删除数据，是否确实删除，是否将不该删除的删除了

（类似更新数据）

1. 异常发生的判断

|  |
| --- |
| /\*\*  \* Case 3. 不存在的藏品编号。  \*/  collectionNum = "A0000002";  collectionRegisterAction.setCollectionNum(collectionNum);  // 告诉JUnit接下来需要抛出异常 ResourceNotFoundException  thrown.expect(ResourceNotFoundException.**class**);  result = collectionRegisterAction.init(); |

* 一些简化代码的方法的使用

|  |
| --- |
| /\*\*  \* JUnit断言工具类  \* **@version** 01-00  \*/  **public** **class** JUnitAssertUtils {  **public** **static** **void** noStrutsErrors(BasicActionSupport action)  **public** **static** **void** hasFieldErrors(BasicActionSupport action, String errKey)  **public** **static** **void** hasFieldErrors(BasicActionSupport action)  **public** **static** **void** equalGivenProperties(Object expectObj, Object actualObj, String... givenProperties)  **public** **static** **void** equal(Object expectObj, Object actualObj, String... excludeProperties)  } |

1. resultJson的比较

以CollectionRegisterActionTest的testSaveBasic1为例：

|  |
| --- |
| ResultJson<Map<String, Object>> rjson = collectionRegisterAction.getResultJson();  JUnitAssertUtils.*equal*(saveBasicResultJson1, rjson); |

注意xml中saveBasicResultJson1如何定义

|  |
| --- |
| <bean id=*"saveBasicResultJson1"* class=*"com.ld.common.web.ResultJson"*>  <property name=*"flag"* value=*"0"*/>  <property name=*"message"* value=*"[藏品的基本信息]的数据登陆成功。"*/>  <property name=*"data"*>  <map>  <entry key=*"registerFlag"* >  <bean class=*"java.lang.Boolean"* >  <constructor-arg>  <value>true</value>  </constructor-arg>  </bean>  </entry>  <entry key=*"basicSaveResortDto"*>  <bean class=*"com.ld.collections.dto.register.CLRGCollectionBasicSaveResultDto"*>  <property name=*"collectionNum"* value=*"A0000001"*/>  <property name=*"registerNum"* value=*"1"*/>  <property name=*"classifyNum"* value=*"C-1"*/>  </bean>  </entry>  </map>  </property>  </bean> |

data部分根据实际情况配置，这里的例子是使用的map

1. 常用断言：

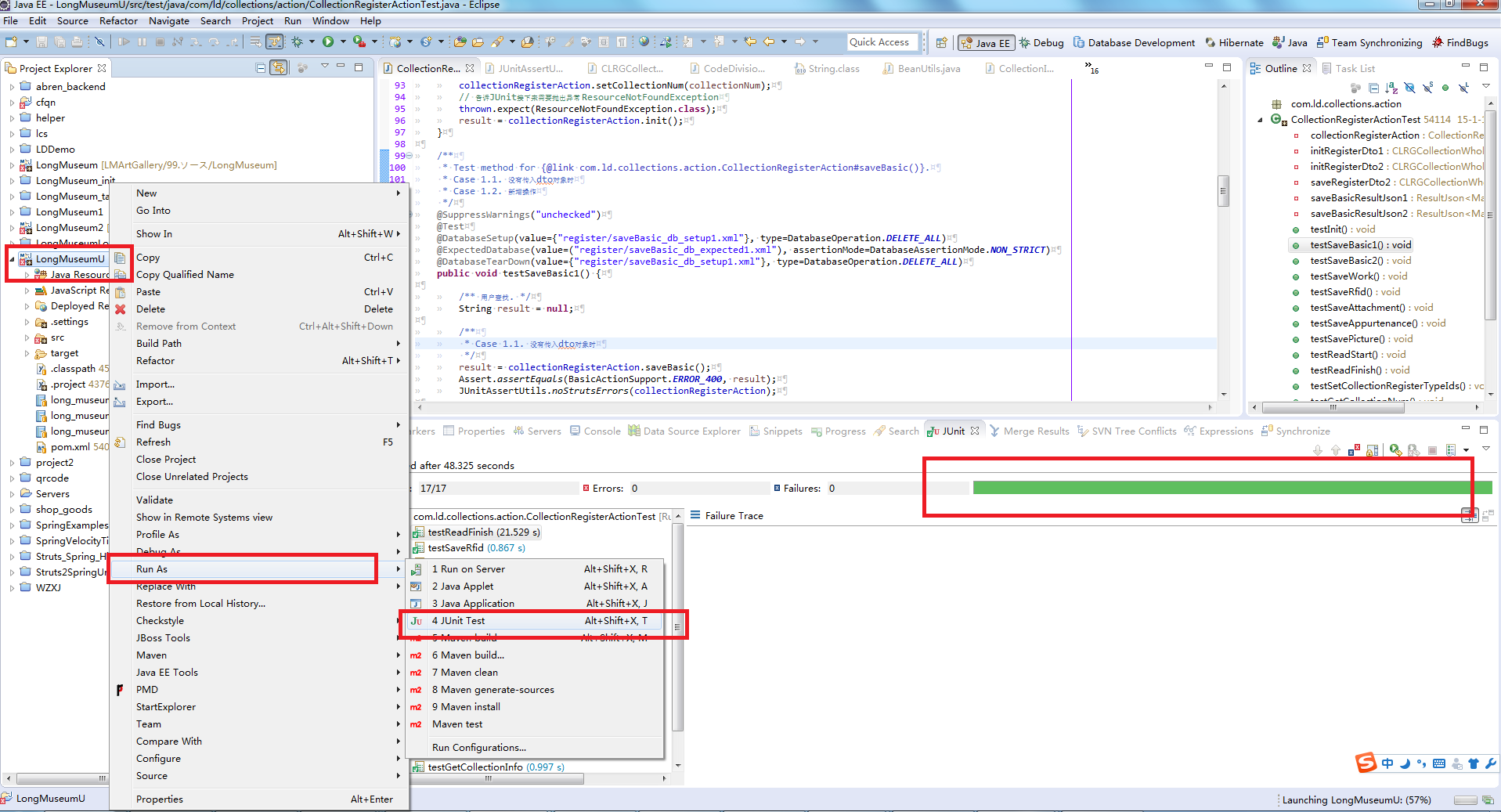
<https://github.com/junit-team/junit/wiki/Assertions>

|  |
| --- |
| import static org.hamcrest.CoreMatchers.allOf;  import static org.hamcrest.CoreMatchers.anyOf;  import static org.hamcrest.CoreMatchers.equalTo;  import static org.hamcrest.CoreMatchers.not;  import static org.hamcrest.CoreMatchers.sameInstance;  import static org.hamcrest.CoreMatchers.startsWith;  import static org.junit.Assert.assertThat;  import static org.junit.matchers.JUnitMatchers.both;  import static org.junit.matchers.JUnitMatchers.containsString;  import static org.junit.matchers.JUnitMatchers.everyItem;  import static org.junit.matchers.JUnitMatchers.hasItems;  import java.util.Arrays;  import org.hamcrest.core.CombinableMatcher;  import org.junit.Test;  public class AssertTests {  @Test  public void testAssertArrayEquals() {  byte[] expected = "trial".getBytes();  byte[] actual = "trial".getBytes();  org.junit.Assert.assertArrayEquals("failure - byte arrays not same", expected, actual);  }  @Test  public void testAssertEquals() {  org.junit.Assert.assertEquals("failure - strings are not equal", "text", "text");  }  @Test  public void testAssertFalse() {  org.junit.Assert.assertFalse("failure - should be false", false);  }  @Test  public void testAssertNotNull() {  org.junit.Assert.assertNotNull("should not be null", new Object());  }  @Test  public void testAssertNotSame() {  org.junit.Assert.assertNotSame("should not be same Object", new Object(), new Object());  }  @Test  public void testAssertNull() {  org.junit.Assert.assertNull("should be null", null);  }  @Test  public void testAssertSame() {  Integer aNumber = Integer.valueOf(768);  org.junit.Assert.assertSame("should be same", aNumber, aNumber);  }  // JUnit Matchers assertThat  @Test  public void testAssertThatBothContainsString() {  org.junit.Assert.assertThat("albumen", both(containsString("a")).and(containsString("b")));  }  @Test  public void testAssertThathasItemsContainsString() {  org.junit.Assert.assertThat(Arrays.asList("one", "two", "three"), hasItems("one", "three"));  }  @Test  public void testAssertThatEveryItemContainsString() {  org.junit.Assert.assertThat(Arrays.asList(new String[] { "fun", "ban", "net" }), everyItem(containsString("n")));  }  // Core Hamcrest Matchers with assertThat  @Test  public void testAssertThatHamcrestCoreMatchers() {  assertThat("good", allOf(equalTo("good"), startsWith("good")));  assertThat("good", not(allOf(equalTo("bad"), equalTo("good"))));  assertThat("good", anyOf(equalTo("bad"), equalTo("good")));  assertThat(7, not(CombinableMatcher.<Integer> either(equalTo(3)).or(equalTo(4))));  assertThat(new Object(), not(sameInstance(new Object())));  }  @Test  public void testAssertTrue() {  org.junit.Assert.assertTrue("failure - should be true", true);  }  } |

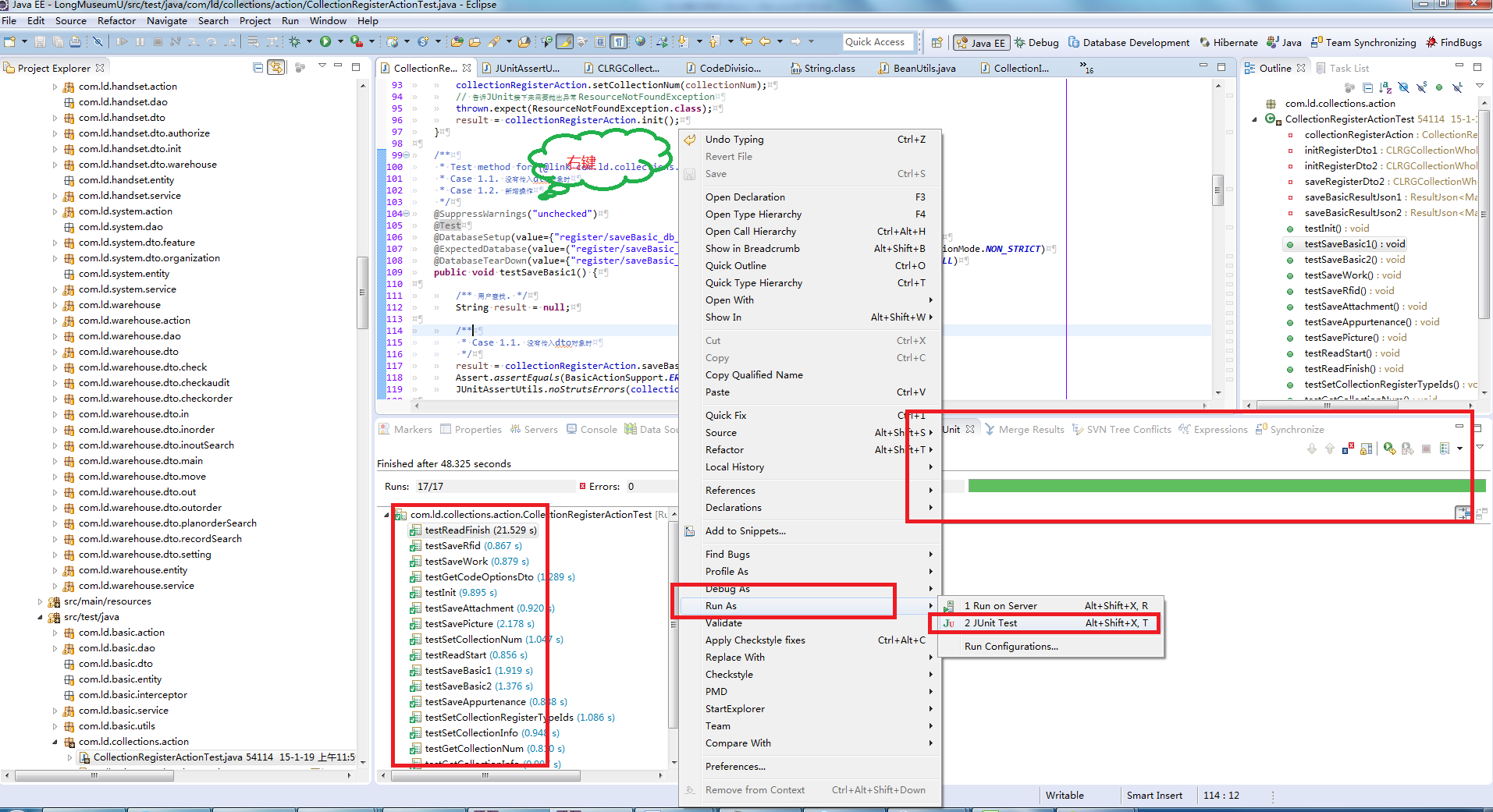
Hamcrest的api

<http://hamcrest.org/JavaHamcrest/javadoc/1.3/>

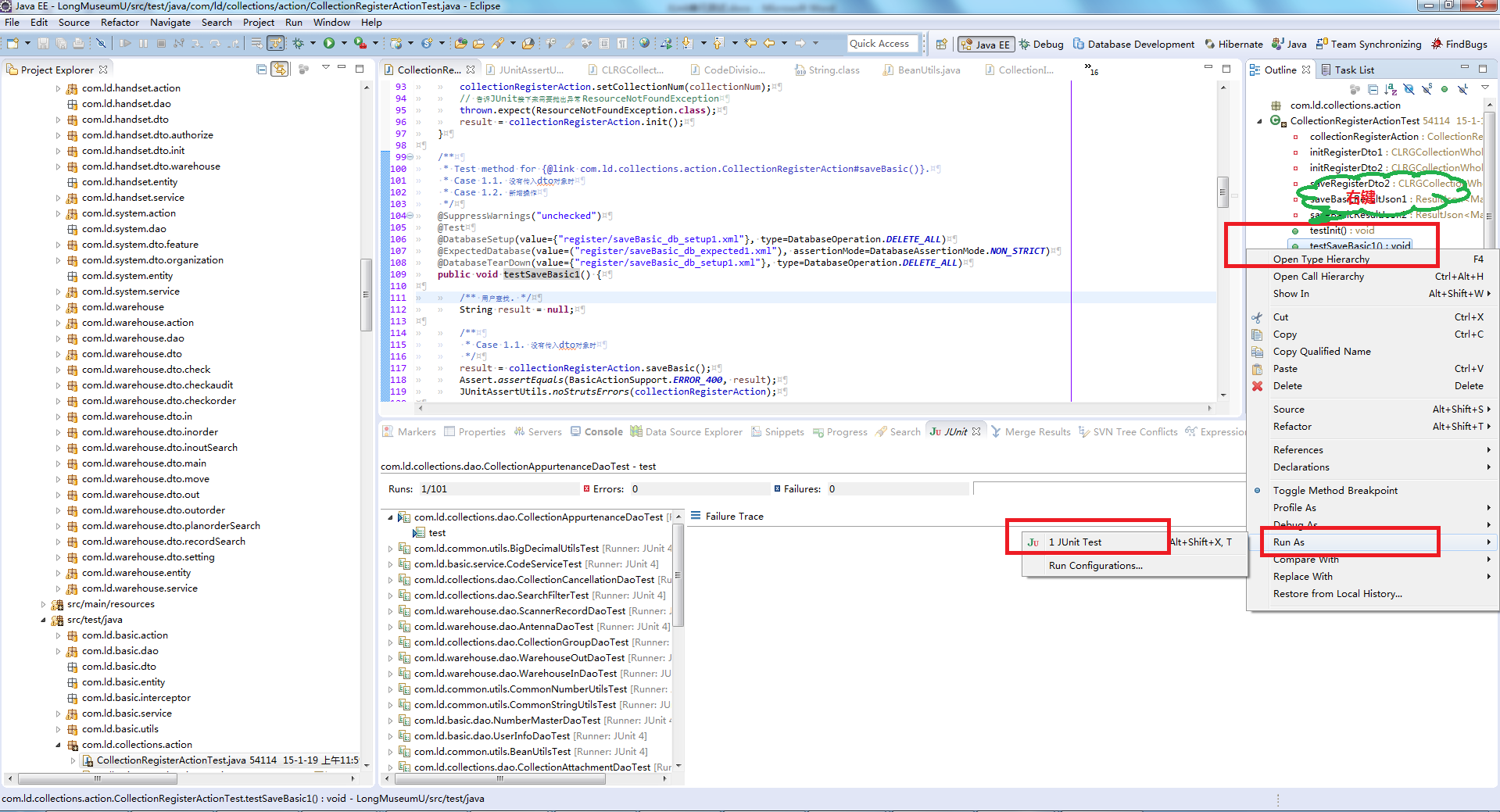
1. 测试用例运行：
   1. Eclipse中
   2. 整个项目（不常用）

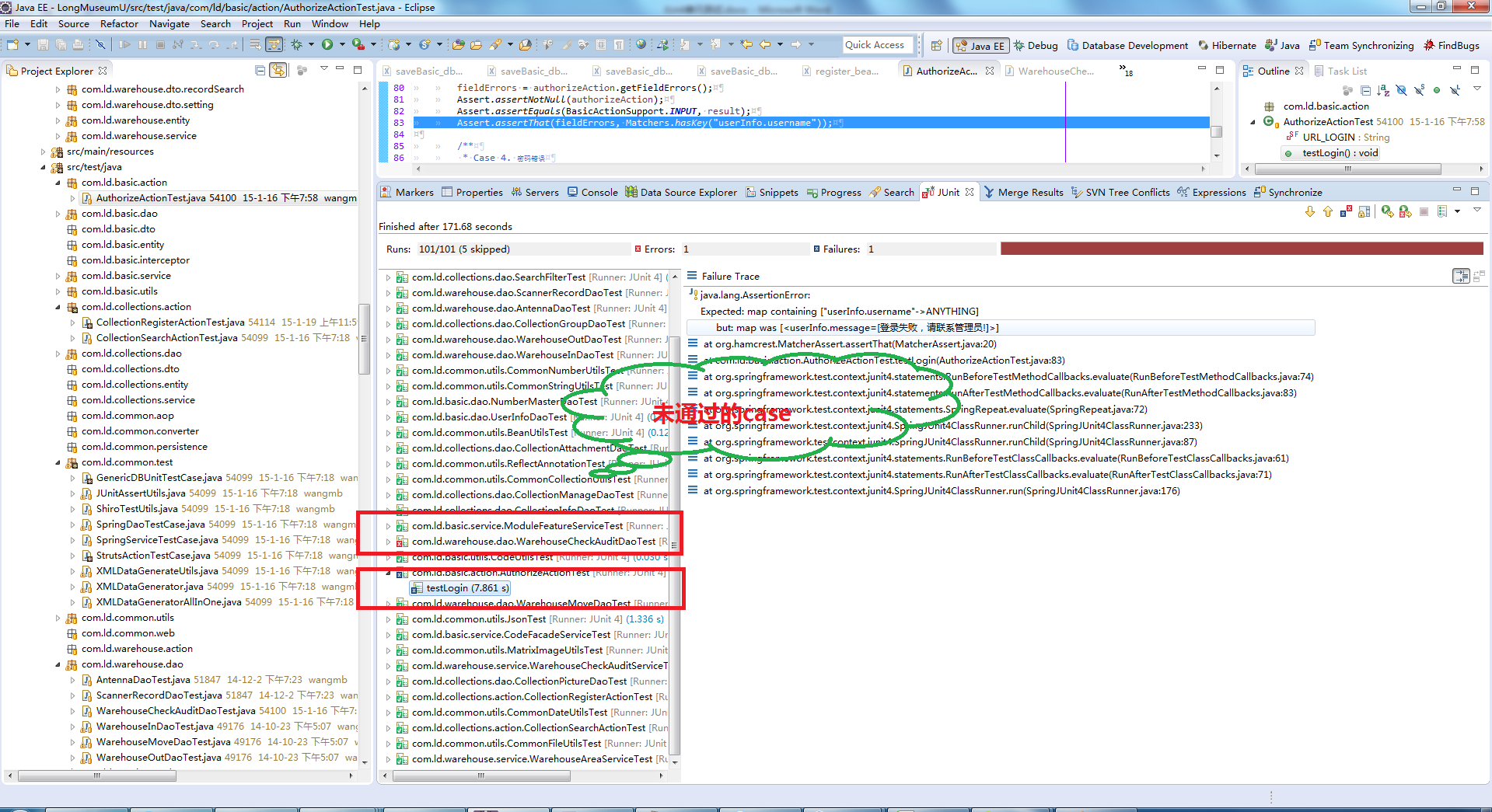


* 1. 单个类（常用）



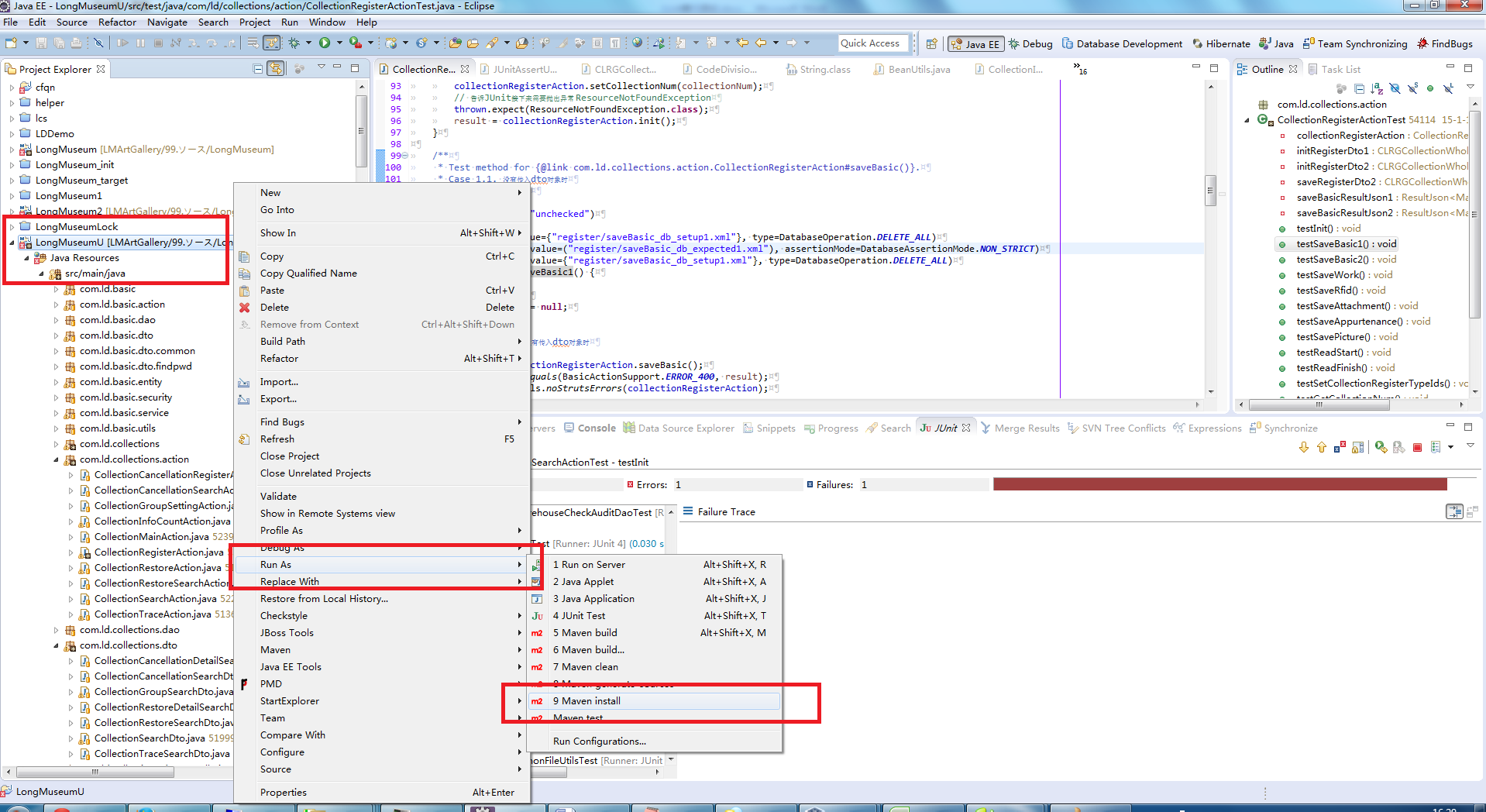
* 1. 单个方法（常用）





* 1. Maven中

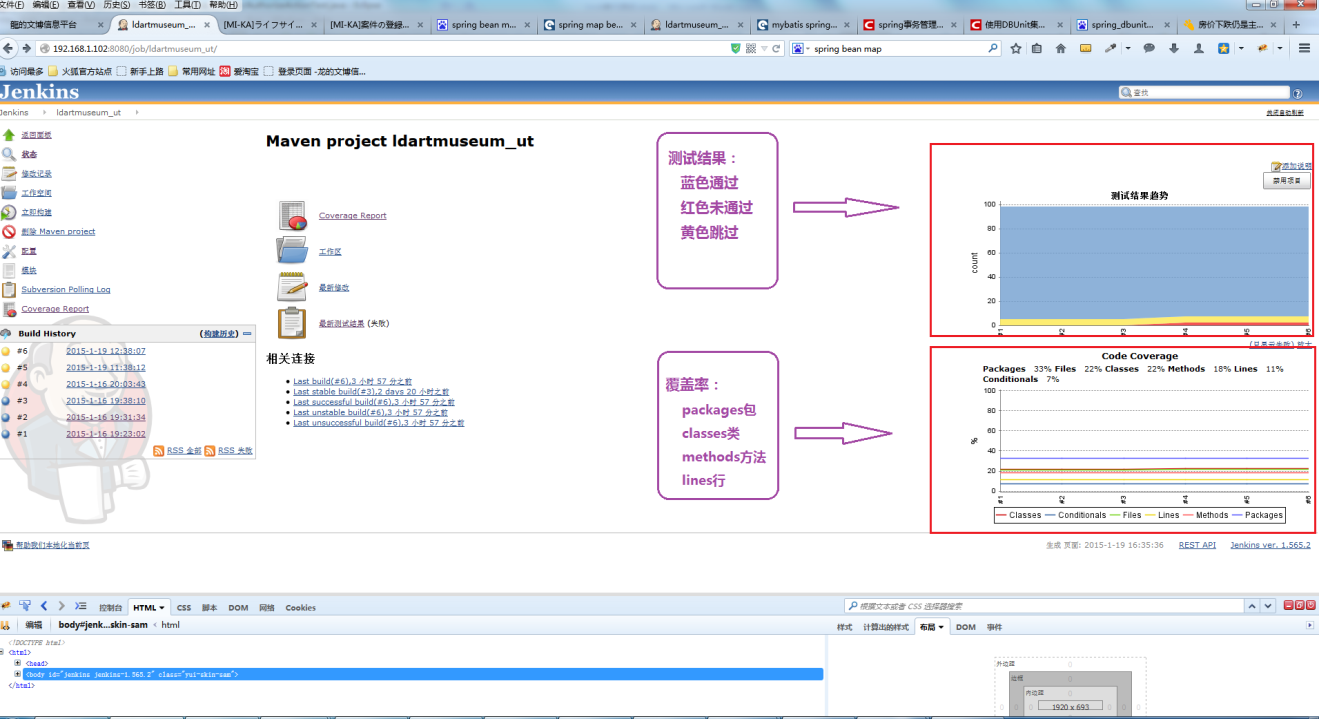
整个项目



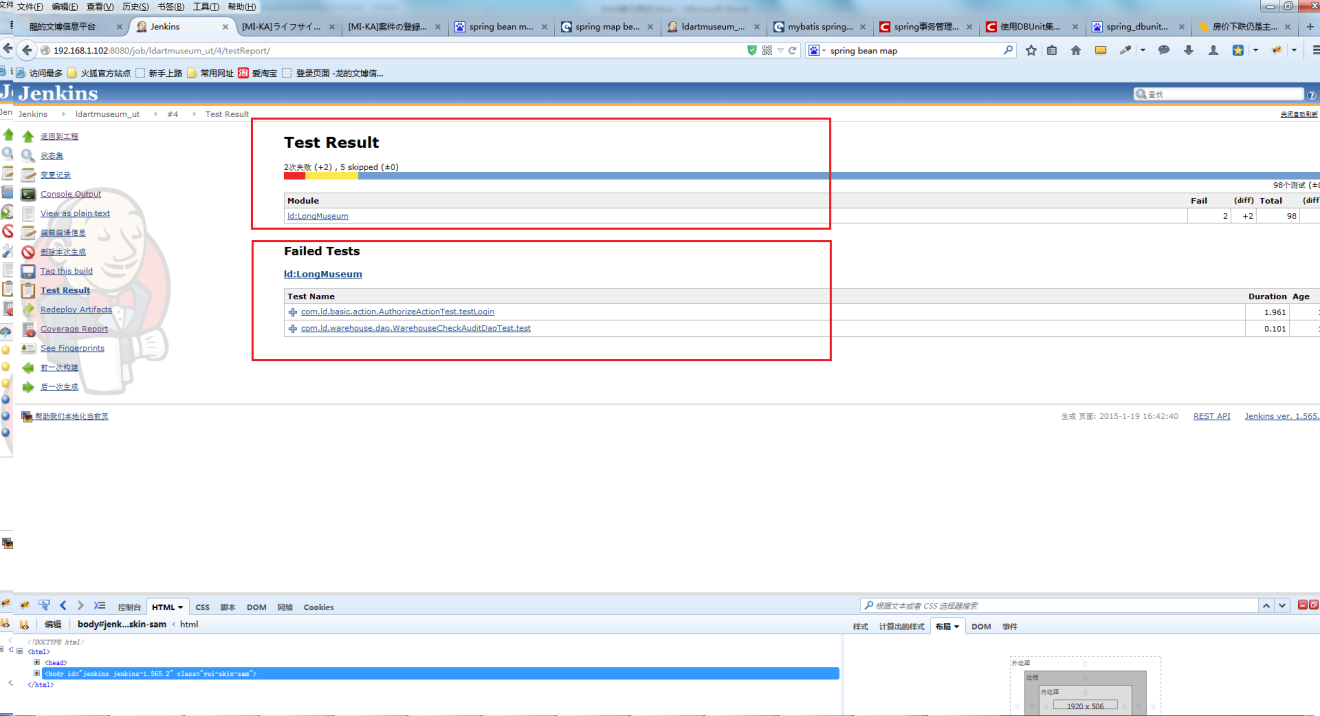
* 1. Jenkins查看覆盖率

<http://192.168.1.102:8080/job/ldartmuseum_ut>

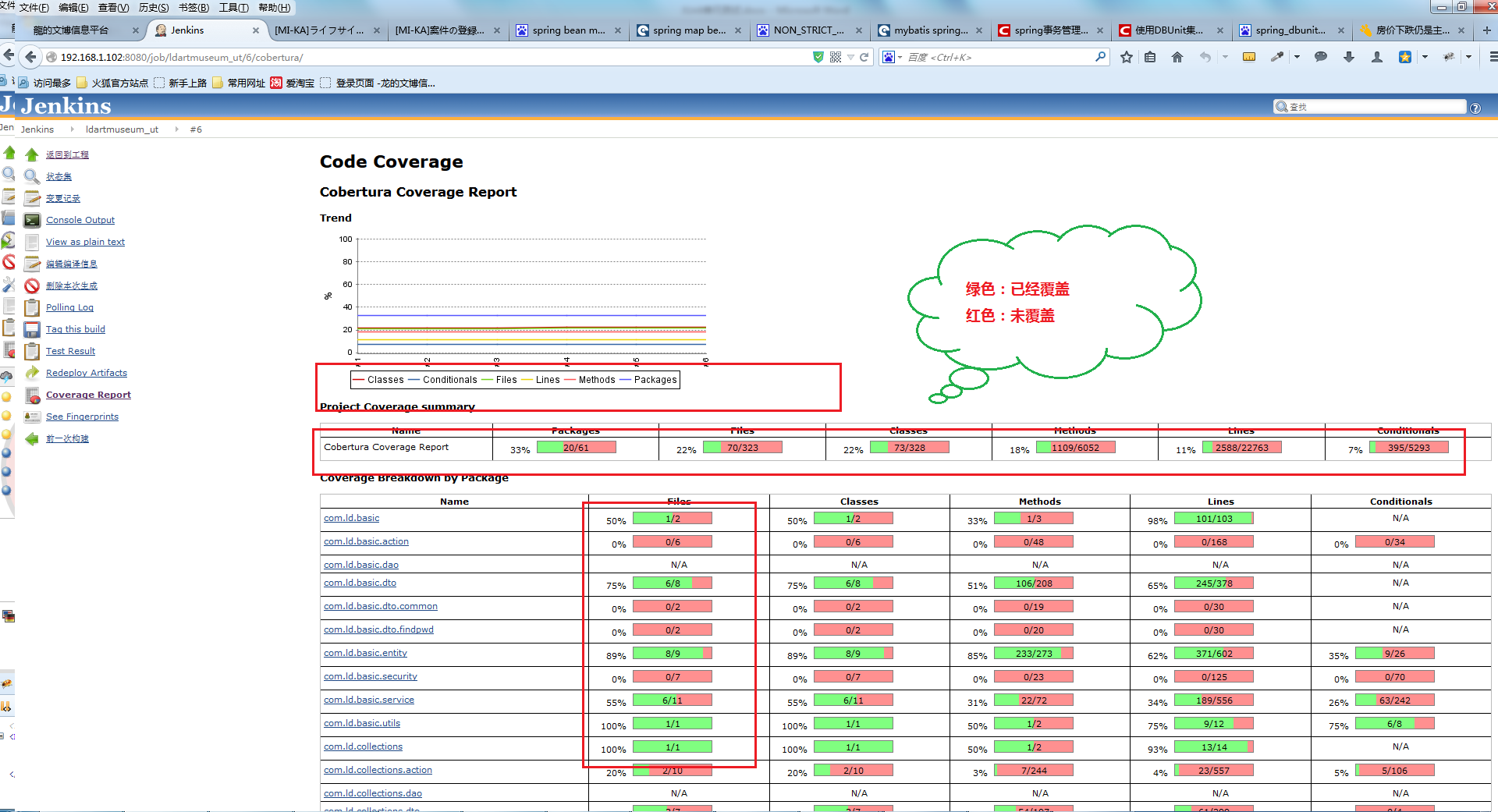
代码上传到svn



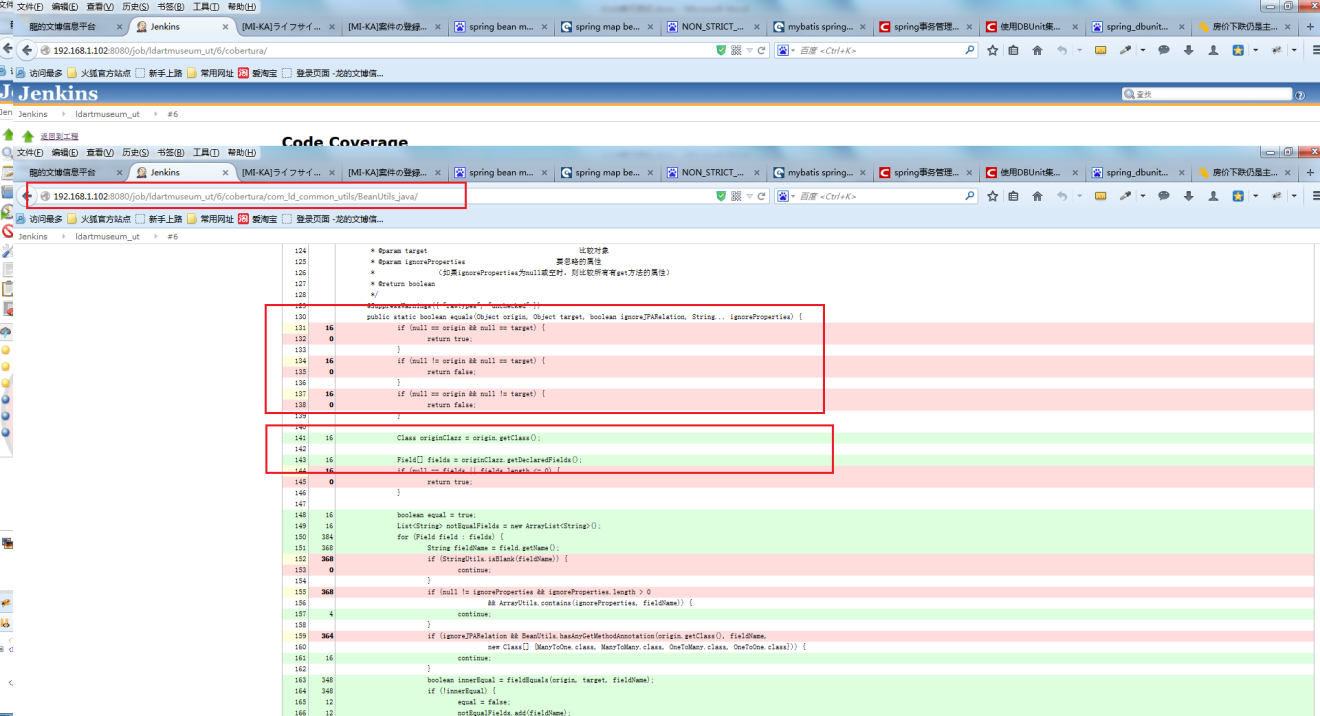
详细的测试结果



详细的覆盖率



单个类的覆盖率



一些注意问题：

1. 测试用例之间不相互影响，测试用例可以重复执行

测试用例1生成的数据不能用于测试用例2

这样就涉及到一个问题数据清理，准备数据

* 1. 数据准备，使用DBUnit帮我们做

测试方法上使用注解

|  |
| --- |
| @DatabaseSetup(value={"register/init\_db\_setup.xml"}) |

* 1. 数据清理

1. 数据更新方法

@DatabaseTearDown(value={"register/saveBasic\_db\_expected.xml"}, type=DatabaseOperation.***DELETE\_ALL***)

1. 数据准备(DatabaseSetup)生成的数据，加上一下注解，清理

@DatabaseTearDown(value={"register/init\_db\_setup.xml"}, type=DatabaseOperation.***DELETE\_ALL***)

init\_db\_setup.xml文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <dataset>  <t001 A100001=*"A0000001"* A100002=*"寿桃"* A100003=*"刘小东《寿桃》布面油彩 200×200cm 2010"*  A100004=*"Л2801Л2802Л2803Л2804Л2805Л"* A160001=*""* A100005=*"0101"*  A100006=*"Л02Л"* A100007=*"Л0201Л0202Л0203Л0204Л0205Л"* A100008=*"Л0301Л0302Л0303Л"*  A100009=*"Л0401Л0402Л"* A100010=*"Л0501Л0502Л0503Л0504Л"* A100011=*"Л2010Л2011Л2014Л2016Л"*  A100012=*"Л0601Л0602Л0603Л0604Л"* A100013=*"Л1域Л2域Л3域Л4域Л"* A100014=*"Л0701Л0702Л0703Л0704Л"*  A100015=*"AU000001"* A100016=*"AU000030"* A100017=*"AU000070"* A100018=*"AU000091"*  A100019=*"0801"* A100020=*"Л01Л02Л03Л04Л"* A100021=*"布面油彩"* A100022=*"0901"*  A100023=*"Л01Л02Л03Л04Л05Л06Л07Л"* A100024=*"A0711 实 际 用 途"*  A100025=*"Л1001Л1002Л1015Л"* A100026=*"A0811 流 派"* A100027=*"A0812 绘 画 技 法"*  A100028=*"A0821 装 饰 生 成 工 艺"* A100029=*"A0822 文字生成工艺"* A100030=*"Л1101Л1102Л1104Л"*  A100031=*"A0911 形 态 特 征"* A100032=*"A0912 独 特 标 志"* A100033=*"1201"*  A100034=*"01"* A100035=*"完 残 状 况"* A100036=*"1301"* A100037=*"A1110 颜色"*  A100038=*"1401"* A100039=*"A1120 光泽"* A100040=*"1501"* A100041=*"A1202 装 饰 题 材 类 别"*  A100042=*"1601"* A100043=*"Л02Л03Л"* A100044=*"A1204 施 饰 部 位"*  A100045=*"A1301 铭 文"* A100046=*"A1302 款 识"* A100047=*"A1303 题 跋"*  A100048=*"A1304 题 名"* A100049=*"A1305 印 鉴"* A100050=*"A1306 内容提要"*  A100051=*"1703"* A100052=*"Л99Л"* A100053=*"A1401 文 字 种 类(其 他)"* A100054=*"1801"*  A100055=*"Л03Л04Л"* A100056=*"A1403 字 迹 颜 色"* A100057=*"A1410 字 体"*  A100058=*"Л1901Л1902Л1903Л"* A100059=*"A1502 模 式 类 别"* A100060=*"A1503 发育阶段"*  A100061=*"A1504 性别"* A100062=*"200×200"* A100063=*"2002"* A100064=*"1621"*  A100065=*"2101"* A100066=*"1631"* A100067=*"2202"* A100068=*"2"* A100069=*"2302"*  A100070=*"1"* A100071=*"2401"* A100072=*"A2002 备 注"* A100073=*"E20034120131F2000EE8B88C060D"*  A100074=*"E2003412013CF2000EE8B89C0611"* A100075=*""* A100076=*""* A100077=*"2909"*  S100003=*"3"* S100009=*"3511"* S100011=*"浦东库房"* C999995=*"false"*  C999991=*"2014-09-17 12:00:00.0"* C999992=*"U9999999"* C999993=*"2015-01-14 16:57:59.0"*  C999994=*"U0000000"* A100078=*"Л植物Л动物Л古人类Л"* />  <t003 A100001=*"A0000001"* A170001=*"2602"* A170002=*"和辉艺术"*  A170003=*"B0103 来 源 号"* A170004=*"出土时间1952"* A170005=*"出土地点"*  A170006=*"发掘人"* A170007=*"B0104 搜集经过"* A170008=*"111.00"* A170009=*"112.00"*  A170010=*"B0121 流传经历"* A170011=*"B0201 收藏单位"* A170012=*"B0211 入 馆 登 记 号"*  A170013=*"2013-01-14"* A170014=*"1"* A170015=*"C-1"* C999995=*"false"*  C999991=*"2014-09-17 12:00:00.0"* C999992=*"U9999999"* C999993=*"2015-01-14 16:58:51.0"*  C999994=*"U0000000"* />  </dataset> |

DBUnit将帮我们生成数据

1. 注意：测试取得的数据是否正确，准备的数据必须每个字段的值必须各不相同
2. Dto类加上注解@Dto，BeanUtils.equal()方法会用到
3. 测试数据库初始化

目前，暂时使用mysql

/LongMuseumU/src/test/resources/datas/db\_test\_init.sql

数据库名字：ldartgallery\_junit

未来会改成纯java数据库 hsqldb

1. 找个专人改下

Dto类加上注解@Dto

Dao和Service的测试基类

JUnit的一些资源：

官网：<http://junit.org/index.html>

一些断言：<https://github.com/junit-team/junit/wiki/Assertions>