**由JDK变更引发EOS 6.0升级到WAS7故障的解决方案**

最近在给客户把EOS 6.0升级到WAS 7时，发现通过手工配置并安装完EOS应用后，登陆到eos-governer后，发现所有获取应用名称以及应用状态的地方都显示为空，但其记录条数正确，在寻找解决的过程中，发现其是由于JDK的升级带来的问题，其定位问题的过程如下供大家参考。

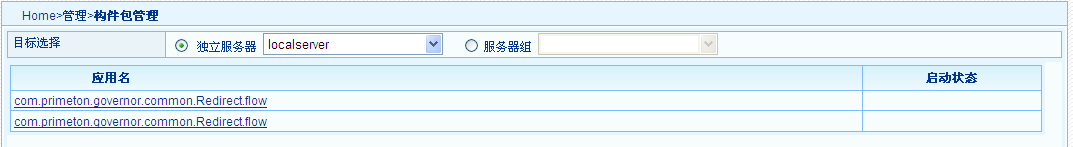
1. **现象**

EOS 6.0由于仅支持WAS6.1的安装，因此只能通过手工的方式进行配置和安装。首先把EOS使用的各应用服务器的参数配置通过WAS控制台进行了一一设置，并把应用部署，启动WAS后应用能正常启动。登陆eos-governor验证一下基本功能，发现出了问题，其应用等信息都为空，但记录条数正确，显示界面如下：

应用管理：



构件包管理：



1. **问题查找**
2. 首先下载了eos-governor工程，找到对应的应用管理页面流ApplicationManage.flow，检查其获取应用的逻辑，跟踪发现其数据处理正常。
3. 逻辑处理没有问题，那重点就在页面上，初步查看其显示由Tag 完成，基本逻辑如下 ：

|  |
| --- |
| <l:iterate property="list" id="apps">  <b:write iterateId='apps' property='name'/>  </l:iterate> |

难道是tag写的有问题，继续调试bean:write tag，发现其获取值的方法为

|  |
| --- |
| XPathLocator.getInstance().getValue |

根据传递的对象，获取其属性的值，跟踪发现对象是正确的，其各属性也正常，但根据传递的属性取得值为空，这说明问题就出在此方法上。

1. 跟踪getValue方法，内部采用的是asm动态解析的方式获取各属性的值，其首先取得属性的类型，然后获取其读写方法，其类为ASMJavaBeanAccessorFactory。

|  |
| --- |
| public IJavaBeanByteCodeAccessor createAccessor(Class clazz) {  。。。。。。  generateInitMethod(cw, IMPLEMENTS\_CLASS\_NAME, beanMetas);  。。。。。  } |

在执行generateInitMethod时出现了空指针异常

|  |
| --- |
| private static void generateInitMethod(ClassWriter cw,  String IMPLEMENTS\_CLASS\_NAME, BeanMetas beanMetas) {  。。。。。  if (entry.getValue().isPrimitive()) {  cv.visitFieldInsn(Constants.GETSTATIC,  。。。。。  } |

entry.getValue()取得的值为空，如果再执行isPrimitive将导致空指针异常，在程序的外层，如果发现出异常，则直接返回了null，而没有把异常进行记录，导致问题查找起来比较困难。

1. 探究entry.getValue()的原因，其集合由类的原数据生成，其生成方式如下：

|  |
| --- |
| private BeanMetas(Class clazz) {  BeanInfo beanInfo = Introspector.getBeanInfo(clazz);;  PropertyDescriptor[] pDescs = beanInfo.getPropertyDescriptors();  for (PropertyDescriptor pDesc : pDescs) {  if (pDesc.getName().equals("class"))  continue;  propertyTypes.put(pDesc.getName(), pDesc.getPropertyType());  。。。。。  } |

从以上代码中可以看到，肯定是PropertyDescriptor的getPropertyType方法返回为空了。

1. **问题确认**

Eos-governor部署在WAS6上是没有问题的，但在WAS7上有问题，并且还是采用JDK自己的类出现了问题，由此可以判断是由于不同版本对于类的属性处理上有差异造成的，WAS 6.1采用的是JDK 1.5，WAS 7 采用的是JDK 1.6 。

为了重现此问题，写了一个简单的 Bean，然后写了个测试类进行不同JDK下进行测试。

Bean代码：

|  |
| --- |
| **public** **class** TestBean {  **private** String name;  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **boolean** isAuth(**int** aaa){  **return** **true**;  }  } |

测试代码：

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  Class clazz = TestBean.**class**;  BeanInfo beanInfo = Introspector.*getBeanInfo*(clazz);  PropertyDescriptor[] pDescs = beanInfo.getPropertyDescriptors();  **for** (PropertyDescriptor pDesc : pDescs) {  System.*out*.println(pDesc.getName() + " -> "+ pDesc.getPropertyType());  }  } |

其测试的结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| SUN JDK 1.5 | SUN JDK 1.6 | IBM JDK 1.5 | IBM JDK 1.6 |
| class -> class java.lang.Class | class -> class java.lang.Class | class -> class java.lang.Class | class -> class java.lang.Class |
| name -> class java.lang.String | name -> class java.lang.String | name -> class java.lang.String | name -> class java.lang.String |
|  |  |  | **auth -> null** |

从以上结果可以看出，在IBM JDK1.6下会多返回一个属性，但其类型为null，由于JDK的不同造成了空指针的产生。

1. **问题解决**

可以在获取属性时添加判断，把此类的属性进行排除即可，修改示例如下。

|  |
| --- |
| private BeanMetas(Class clazz) {  BeanInfo beanInfo = Introspector.getBeanInfo(clazz);;  PropertyDescriptor[] pDescs = beanInfo.getPropertyDescriptors();  for (PropertyDescriptor pDesc : pDescs) {  if (pDesc.getName().equals("class"))  continue;  **if (pDesc instanceof IndexedPropertyDescriptor) {**  **continue;**  **}**  propertyTypes.put(pDesc.getName(), pDesc.getPropertyType());  。。。。。  } |

此案例可以发现，各厂商的 JDK实现还是有差别的，任何一次升级都可能引起不可预知的错误，这了要求我们把异常的处理机制进一步完善，加快问题的定位速度。