>> 返回首页

探查 ClassCastException

问题描述
ClassCastException 是 JVM 在检测到两个类型间的转换不兼容时引发的运行时异常。此类错误通常会终止用户请求。本模式试图为您提供了解和排除 ClassCastException 错误最常见成因的一些基本要素。

故障排除
请注意，并非下面所有任务都需要完成。有些问题仅通过执行几项任务就可以解决。

快速链接：

为什么发生此问题？
在执行几乎任何子系统（Web 容器、EJB、JCA、群集等）的应用程序代码或 WebLogic Server 代码内均可能发生 ClassCastException。通过转换，可以指示 Java 编译器将给定类型的变量作为另一种变量来处理。对基础类型和用户定义类型都可以进行转换。Java 语言规范定义了允许的转换，其中的大多数可在编译时进行验证。不过，某些转换还需要运行时验证。如果在此运行时验证过程中检测到不兼容，JVM 就会引发 ClassCastException。

假设有一个 S 类型的对象，我们想把它转换为 T 类型。

```
S s;
...
T t = (T) s;
```

如果存在以下情况，上述转换就可能引发 ClassCastException

- S 与 T 不兼容。规范规定：“当应用程序代码尝试将某一对象转换为某一子类时，如果该对象并非该子类的实例，JVM 就会抛出 ClassCastException。”以下是一个示例代码，执行该代码时将会引发此类错误：

```
public class TestCCE {

  public static void main(String args[]) {
    Object obj = new Object();
    String s = (String) obj;
  }
}
```

- S 类型和 T 类型兼容，但加载时使用了不同的 ClassLoader。

  第二个原因实际上是这种错误最常见的原因。这种情况在诊断上有相当的难度，而且需要对 Java 类加载以及 WebLogic 类加载体系结构方面的基础知识有一定程度的了解。

返回页首

什么是 ClassLoader?

ClassLoader 是允许 JVM 查找和加载类的一种 Java 类。JVM 有内置的 ClassLoader。不过，应用程序可以定义自定义 ClassLoader。应用程序定义新的 ClassLoader 通常有两个主要目的：

- 自定义和扩展 JVM 加载类的方式。例如：增加对新的类库（网络、加密文件等）的支持
- 划分 JVM 名称空间，避免名称冲突。举例来说，利用划分技术可同时运行同一应用程序的多个版本（基于空间的划分）。此项技术在应用程序服务器（如 WebLogic Server）内的另一个重要用途是启用应用程序热重新部署，即在不重新启动 JVM 的情况下启动应用程序的新版本（基于时间的划分）。

ClassLoader 按层级方式进行组织。除系统 Boot ClassLoader 外，其它 ClassLoader 都必须有父 ClassLoader。http://e-docs.bea.com/wls/docs81/programming/classloading.html (English) 中提供了对 WebLogic 类加载体系结构的说明。

理解类加载的关键

记住以下内容会有帮助：

- 永远无法在同一 ClassLoader 中重新加载类："热重新部署"需要使用新的 ClassLoader。每个类对其 ClassLoader 的引用都是不可变的：`this.getClass().getClassLoader()`
- 在加载类之前，ClassLoader 始终会先询问其父 ClassLoader（委托模型）。这意味着将永远无法重写"核心"类
- 同级 ClassLoader 间互不了解
- 由不同 ClassLoader 加载的同一类文件也会被视为不同的类，即便每个字节都完全相同。这是 ClassCastException 的一个典型成因。
- 可以使用 `Thread.setContextClassloader(cl)` 将 ClassLoader 连接到线程的上下文

返回页首

诊断

通常可以在服务器的日志和/或客户端获得完整的 ClassCastException 堆栈跟踪。该堆栈应能使您对涉及的 WebLogic Server 子系统的情况有相当深入的了解。请根据这些信息确认该问题是否与某个已知 WebLogic Server 问题的情况相符。如果相符，请使用相应的解决办法。

如果错误与所有已知问题的情况均不相符，则需要做进一步探查。如果错误出现在您可以编辑和编译源代码的类中，以下方法可能有助于您理解该问题。

假设出现错误的代码行类似于：

```
oo f = (Foo) bar.method();
```

如果按照转换规则该转换应该有效，但仍然引发了 ClassCastException，则可能的情况是：类"bar"和"Foo"是由不同的 ClassLoader 加载的。要验证这一点，请像下面这样拆分该代码行：

```
Object o = bar.method();
System.err.println("The object " + o + " classloader is " +
o.getClass().getClassLoader());
System.err.println("Class Foo class loader is " +
Foo.class.getClassLoader());
Foo f = (Foo) o;
```

典型的输出可能与此类似：

```
The object Foo@@3e86d0 classloader is
```

```
sun.misc.Launcher$AppClassLoader@b9d04
Class Foo classloader is
weblogic.utils.classloaders.ChangeAwareClassLoader@5998cb finder:
weblogic.utils.classloaders.MultiClassFinder@7c2528
```

下一步是探查为什么涉及了不同的 ClassLoader。请执行下列检查清单中的各项检查：

- 检查应用程序打包情况，并检查“Foo”是否是使用由不同 ClassLoader 加载的不同模块打包而成。在这方面众所周知的一个成因是 Web 应用程序的“prefer-web-inf-classes”功能（请参阅已知的 WebLogic 问题）
- 检查它正在使用的应用程序代码或某个框架代码是否更改了 WebLogic 线程的上下文 ClassLoader，且在请求流程期间未将其还原。如果应用程序代码内有对 `Thread.setContextClassloader(cl)` 的调用，就可能存在这种情况。
- 如果上述所有措施均无济于事，请尝试将问题隔离在一个简单的、可重现的测试案例中，然后联系 BEA 技术支持部门，以做进一步探查。

返回页首

已知的 WebLogic Server 问题

以下汇集了可导致 ClassCastException 错误的各种情况，您在使用各种 WebLogic 子系统时可能会遇到这些情况。

小程序
有关该情况的完整说明，请参考 http://e-docs.bea.com/wls/docs81/applets/usingapplets.html (English)

摘要：如果小程序中的 WebLogic Server 客户端尝试从 ClassLoader 获取某些资源信息，且一并使用了缓存标志和 `codebase=/bea_wls_internal/classes` 标志，就可能会抛出 `ClassCastException`。

解决方法：

- 在将类路径 servlet 用作代码基时，请不要使用“cache_option”和“cache_archive”一类的缓存选项。
- 使用缓存选项时，请不要使用 /classes (ClasspathServlet) 做为代码基。如果要这样做，请先使用归档实用程序打包客户端 JAR。

有关此限制的详细信息，请参阅 http://developer.java.sun.com/developer/bugParade/bugs/4648591.html (English)。

返回页首

JCA Connector

有关该情况的完整说明，请参考 http://e-docs.bea.com/wls/docs81/notes/issues.html (English)。

摘要：发出连接请求时，WebLogic Server 会返回一个代理对象，该对象通过资源适配器将连接对象封装后返回到客户端。WebLogic Server 使用该代理来提供一些功能，帮助应用程序使用 WebLogic Server 的“J2EE 连接器体系结构”实现。这些功能包括 (1) 连接泄漏检测功能和 (2) 连接请求在启动使用该连接的全局事务之前发出时，推迟 XAResource 登记。

如果将连接请求返回的连接对象向原始的 `Connection` 类进行了转换，就可能发生 `ClassCastException`。导致该异常的对象不外乎在连接请求过程中：(1) 资源适配器进行转换时或 (2) 客户端进行转换时。

在 WebLogic Server 8.1 SP2 中，尝试检测由上述资源适配器情况 (1) 导致的 `ClassCastException`。如果服务器检测到该转换失败，将关闭代理包装器功能，并在连接请求期间返回连接对象（不进行包装）。服务器会记录一条警告消息，说明代理包装器已被关闭。出现此类转换故障时，连接泄漏检测和 XAResource 推迟登记功能也将被关闭（但当前在控制台监视中并不会就此给出任何指示）。

WebLogic Server 尝试以使用容器管理的安全性的客户端身份检测 `ClassCastException`。如果要这样做，则部署的资源适配器须定义安全性 Credential。

如果客户端在执行转换时发生 `ClassCastException`，可按如下方式修改客户（客户端）代码：

解决方法：如果客户端将连接对象转换为 `MyConnection`，而不是将 `MyConnection` 作为实现资源适配器的 `Connection` 接口的一个类，请将该对象修改为一个扩展 `Connection` 的接口。实现一个用于实现 `MyConnection` 接口的 `MyConnectionImpl` 类。

返回页首

Servlet 动态重新加载

有关该情况的完整说明，请参考 http://e-docs.bea.com/wls/docs81/jsp/reference.html (English)

摘要：要在会话过程中动态重新加载 servlet 或 JSP，servlet 会话中存储的对象必须是可序列化的。需要进行序列化是因为，servlet 是使用新的 ClassLoader 重新加载的，而这会导致此前加载的所有类（旧版本 servlet 中的类）与使用新的 ClassLoader 加载的所有类（新版本 servlet 类）发生不兼容的情况。这种不兼容会导致 servlet 返回 `ClassCastException` 错误。

使用 Prefer-web-inf-classes 功能

有关该情况的完整说明，请参考 http://e-docs.bea.com/wls/docs81/programming/classloading.html (English)

摘要：weblogic.xml Web 应用程序部署描述符包含一个 prefer-web-inf-classes 元素（container-descriptor 元素的子元素）。缺省情况下，此元素设置为 False。如果将此元素设置为 True，则不遵循 ClassLoader 委托模型，从而使 Web 应用程序中的类定义的加载顺序优先于更高级别的 ClassLoader 中的类定义。这样 Web 应用程序就可使用其自己版本的第三方类，该类也可能是 WebLogic Server 的一部分。请参阅 weblogic.xml Deployment Descriptor Elements (English)。

使用该功能时，必须注意不要混淆使用 Web 应用程序的类定义创建的实例与使用服务器的定义创建的实例。如果混淆了此类实例，就会发生 ClassCastException。

群集：在 http 会话中存储包含 EJBObject 的自定义对象 - CR102119

摘要：可序列化的自定义对象会包装 EJBObject（对 EJB 的引用）。该自定义对象存储在 http 会话中。会话复制时，这种设计就会导致 ClassCastException。

解决方法：在以后版本的 WebLogic Server 中将彻底解决该问题。目前的解决办法是在包装器内存储 EJB 句柄（而不是 EJBObject）。

返回页首

群集：Failover 后使用 http 会话中的 EJB 句柄 - CR187062

摘要：在群集中分别部署 webApp 和 EJB。EJB 句柄包装在可序列化的自定义对象中，而该对象存储在 http 会话中。Failover 后，通过句柄访问 EJB 就会发生 ClassCastException。

解决方法：在同一个 ear 文件中包装 webApp 和 EJB。该问题存在于 WLS 8.1sp2 中，目前正在等待更正。

返回页首

需要更多帮助？

如果您已经理解这个模式，但仍需要更多帮助，您可以：

1. 在 http://support.bea.com/ 上查询 AskBEA（例如，使用“ClassCastException”），以查找其它已发布的解决办法。
2. 在 http://forums.bea.com/ 上，向 BEA 的某个新闻组提出更详细具体的问题。

如果这还不能解决您的问题，并且您拥有有效的技术支持合同，您可以通过登录以下网站来打开支持案例：http://support.bea.com/。

反馈

请给我们提供您的意见，说明此支持诊断模式**“探查 ClassCastExceptions”**一文是否有所帮助、您需要的任何解释，以及对支持诊断模式的新主题的任何要求。

返回页首