JAVA基础：四种Java脚本语言之评测 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_JAVA_E5_9F_BA_E7_A1_80_c104_145500.htm 一、脚本解释器概述 在一些Java应用的需求中，集成某种脚本语言的支持能够带来很大的方便。例如，用户可能想要编写脚本程序驱动应用、扩展应用，或为了简化操作而编写循环和其他流程控制逻辑。在这些情况下，一种理想的解决方案是在Java应用中提供对脚本语言解释器的支持，让脚本语言解释器读取用户编写的脚本并在应用提供的类上运行这些脚本。为了实现这个目标，你可以在Java应用所运行的JVM中，运行一个基于Java的脚本语言解释器。 一些支持库，例如IBM的Bean Scripting Framework，能够帮助你把不同的脚本语言集成到Java程序。这些支持框架能够让你的Java应用在不作大量修改的情况下，运行Tcl、Python和其他语言编写的脚本。 在Java应用中集成了脚本解释器之后，用户编写的脚本能够直接引用Java应用的类，就如这些脚本属于Java程序的一部分一样。这种思路既有优点也有缺点。其优点在于，如果你想要用脚本驱动的方式对应用进行回归测试，或者想要通过脚本对应用进行低级调用，它能够带来很大的方便；其缺点在于，如果用户的脚本直接操作Java程序的内部结构而不是经过认可的API，它可能影响Java程序的完整性和应用的安全。因此，应当仔细地规划那些允许用户针对其编写脚本的API，并声明程序的其余部分不允许用脚本操作。另外，你还可以对那些不想让用户针对其进行脚本编程的类和方法名称进行模糊处理，只留出那些允许脚本编程的API类和方法名字。这样，你就能够有效地降低

喜欢冒险的用户直接用脚本操作受保护的类和方法的可能性。 在Java程序中支持多种脚本语言有着非同寻常的意义，但如果你正在编写的是一个商业应用，则应当慎重考虑尽管你为用户提供了最完善的功能，但同时也带来了最多的出错机会。必须考虑到配置和管理问题，因为至少有一部分的脚本解释器在定期地进行升级和更新，这样你就必须花很大的力气管理各个解释器的哪些版本适合于Java应用的哪些版本。如果用户为了解决旧脚本解释器中存在的BUG，对其中某个脚本解释器进行了升级，你的Java应用就会运行在一种未经完全测试的配置下。数天或数星期之后，用户也许会发现由于脚本引擎升级而产生的问题，但他们很可能不会把脚本引擎升级的事情告诉你，这时你就很难再次重复试验出用户报告的错误了。 另外，用户很可能坚持认为你必须为Java应用支持的脚本解释器提供补丁。一些脚本解释器按照源代码开放的模式及时进行维护和更新；对于这些脚本解释器，可能有专家帮助你解决问题、修补解释器，或在新的发行版中引入补丁。这是很重要的，因为脚本解释器是一个很复杂的工具，包含大量的代码，如果没有专家的支持，对于自己修改脚本解释器这一令人烦恼的任务，你很可能束手无策。 为了避免出现这种问题，你应该对于每一种准备在Java应用中提供支持的脚本解释器进行全面的测试。对于每一种解释器，确保它能够顺利地处理绝大多数常见的使用情形，确保它即使在极端苛刻的条件下运行大量的脚本也不会出现大的内存漏洞，确保当你对Java程序和脚本解释器进行严格的Beta测试时不会出现任何意外的情况。当然，这种前期测试需要投入时间和其他资源；但不管怎样，测试投入总是物有所值的。 二、保

持系统简洁 如果你必须在Java应用中提供脚本支持，首先必须选择一个最符合应用要求和用户基础的脚本解释器。选择合适的解释器能够简化集成解释器的代码，减少客户支持方面的支出，以及提高应用的稳定性。最困难的问题在于：如果只能选用一种解释器，应该选用哪一种呢？ 我比较了几种脚本解释器，开始时考虑的脚本语言包括Tcl、Python、Perl、JavaScript和BeanShell。接着，在深入分析之前，我放弃了Perl。为什么呢？因为Perl没有用Java写的解释器。假设你选择了一个用本机代码实现的脚本解释器，例如Perl，则Java应用和脚本代码之间的交互就不再直接进行；另外，对于每一个你想要支持的操作系统，都必须提供一个脚本解释器的二进制代码库。由于许多开发者选择Java是因为看中了它的跨平台可移植性，为了保证Java应用有这种优点，所以最好选择一种不依赖于本机代码的解释器。和Perl不同，Tcl、Python、JavaScript和BeanShell都有基于Java的解释器，所以这些语言的代码可以与Java应用在同一个JVM和进程之内运行。 基于以上标准，参与本文评测的脚本解释器包括： Jacl：Tcl的Java实现。 Jython：Python的Java实现。 Rhino：JavaScript的Java实现。 BeanShell：一个用Java编写的Java源代码解释器。 限定了待比较的解释器种类之后，接下来就可以从各个方面对它们进行比较了。 三、评测之一：可用性 第一个评测项目是可用性。这项评测分析了是否存在某种解释器不可用的情形。用每一种语言各编写一个简单的测试程序，然后分别用相应的解释器运行，结果发现，所有解释器都通过了测试，每一种解释器都能够稳定地工作或能够方便地与之交互。既然每一种解释器都值得考虑，那么，有哪些因素可能使开发者偏爱

其中一种呢？ 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com