为你的应用程序添加动态Java代码（二） 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__E4_B8_BA_E4_BD_A0_E7_9A_84_E5_c104_145073.htm 实现动态代码的四个步骤让我来揭示在这种表象后面到底发生了什么。基本上，构成动态代码分四个步骤： 部署选择的部分源代码并监控源代码文件的变化 在运行时编译java代码 在运行时装载/重装载java类 将最新的类链接给它的调用者部署选择的部分源代码并监控源代码文件的变化在开始写一些动态的代码之前，我们必须回答的第一个问题是，“ 哪部分代码应该是动态的整个应用程序还是仅仅某些类？ ”从技术上来讲，这部分是没什么约束的。你可以在运行时装载/重装载任何java类。但是在更多的情况下，只有部分代码需要这种灵活性。Postman的例子示范了一种典型的选择动态类的模式。不管系统是如何构成的，最终，总有诸如服务，子系统，组件这样的组装块。这些组装块相对独立，通过预先定义的接口，互相暴露出了自己的功能。在接口后面，是自由变化的执行程序，只要它符合接口定义的限制。这是我们所需要的动态类的明确的性质。简单说来就是：选择实现类作为动态类。文章的其余部分，我们做如下关于选择动态类的假设： 被选择的实现了的java接口从而暴露出自己的功能。 被选择的动态类的执行程序不保留任何关于其客户的状态信息（类似无状态的会话bean），这样动态类的实例可以互相替换。请注意这种假设不是必要的。这样做只是为了让动态代码的实现稍简单一些，以便我们可以集中更多的精力到概念和机制上去。利用心中选定好的动态类，配置源码是很简单的任务。图1指出

了Postman例子的文件结构。Figure 1. The file structure of the Postman example我们知道“ src ”是源码，“ bin ”是二进制码。一件值得注意的事情是动态代码目录，它包括了动态类的源文件。这儿的例子中，只有一个文件PostmanImpl.java。bin和动态代码的目录是需要用来运行应用程序的，src在配置时是不需要的。检查文件改变可以通过比较修改时间和文件大小实现。我们的例子中，对PostmanImpl.java的检查是每次调用一个基于Postman接口的方法。要不，你也可以产生一个后台线程来有规则地检查文件改变。这可能会为大规模的应用程序带来更好的性能。运行时编译java代码探测到源码被改变后，我们要面临编译的问题。将实际的编译工作委派给某个已经存在的java编译器的话，运行时编辑就不成问题了。许多java编译器都是可用的，但在本文中，我们使用包含在Sun的 java SE平台的javac编译器（java SE是Sun为J2SE的新命名）。最简单的方式，你可以仅用一条语句编译java文件,这需要系统在类路径上包含javac编译器的tools.jar（你可以在/lib/下找到tools.jar）：int errorCode = com.sun.tools.javac.Main.compile(new String[] { "-classpath", "bin", "-d", "/temp/dynacode_classes", "dynacode/sample/PostmanImpl.java" }). 