JDK1.6的九大新特性Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/641/2021_2022_JDK16_E7_9A_84_E4_c104_641960.htm 一：Desktop类和SystemTray类 在JDK1.6中，AWT新增加了两个类：Desktop和SystemTray。前者可以用来打开系统默认浏览器浏览指定的URL，打开系统默认邮件客户端给指定的邮箱发邮件，用默认应用程序打开或编辑文件(比如，用记事本打开以txt为后缀名的文件)，用系统默认的打印机打印文档；后者可以用来在系统托盘区创建一个托盘程序。 二：使用JAXB2来实现对象与XML之间的映射 JAXB是Java Architecture for XML Binding的缩写，可以将一个Java对象转变成为XML格式，反之亦然。 我们把对象与关系数据库之间的映射称为ORM，其实也可以把对象与XML之间的映射称为OXM(Object XML Mapping)。原来JAXB是Java EE的一部分，在JDK1.6中，SUN将其放到了Java SE中，这也是SUN的一贯做法。JDK1.6中自带的这个JAXB版本是2.0，比起1.0(JSR 31)来，JAXB2(JSR 222)用JDK5的新特性Annotation来标识要作绑定的类和属性等，这就极大简化了开发的工作量。 实际上，在Java EE 5.0中，EJB和Web Services也通过Annotation来简化开发工作。另外，JAXB2在底层是用StAX(JSR 173)来处理XML文档。除了JAXB之外，我们还可以通过XMLBeans和Castor等来实现同样的功能。 三：理解StAX StAX(JSR 173)是JDK1.6.0中除了DOM和SAX之外的又一种处理XML文档的API。 StAX 的来历：在JAXP1.3(JSR 206)有两种处理XML文档的方法：DOM(Document Object Model)和SAX(Simple API for XML)。 由于JDK1.6.0中的JAXB2(JSR

222)和JAX-WS 2.0(JSR 224)都会用到StAX所以Sun决定把StAX加入到JAXP家族当中来，并将JAXP的版本升级到1.4(JAXP1.4是JAXP1.3的维护版本)。JDK1.6里面JAXP的版本就是1.4。StAX是The Streaming API for XML的缩写，一种利用拉模式解析(pull-parsing)XML文档的API.StAX通过提供一种基于事件迭代器(Iterator)的API让程序员去控制xml文档解析过程，程序遍历这个事件迭代器去处理每一个解析事件，解析事件可以看做是程序拉出来的，也就是程序促使解析器产生一个解析事件然后处理该事件，之后又促使解析器产生下一个解析事件，如此循环直到碰到文档结束符；SAX也是基于事件处理xml文档，但却是用推模式解析，解析器解析完整个xml文档后，才产生解析事件，然后推给程序去处理这些事件；DOM采用的方式是将整个xml文档映射到一颗内存树，这样就可以很容易地得到父节点和子结点以及兄弟节点的数据，但如果文档很大，将会严重影响性能。 四：使用Compiler API 现在我 们可以用JDK1.6 的Compiler API(JSR 199)去动态编译Java源文件，Compiler API结合反射功能就可以实现动态的产生Java代码并编译执行这些代码，有点动态语言的特征。这个特性对于某些需要用到动态编译的应用程序相当有用，比如JSP Web Server，当我们手动修改JSP后，是不希望需要重启Web Server才可以看到效果的，这时候我们就可以用Compiler API来实现动态编译JSP文件，当然，现在的JSP Web Server也是支持JSP热部署的，现在的JSP Web Server通过在运行期间通过Runtime.exec或ProcessBuilder来调用javac来编译代码，这种方式需要我们产生另一个进程去做编译工作，不够优雅而且容易使代码依赖与特定的操作系统；Compiler

API通过一套易用的标准的API提供了更加丰富的方式去做动态编译，而且是跨平台的。 五：轻量级Http Server API JDK1.6提供了一个简单的Http Server API，据此我们可以构建自己的嵌入式Http Server，它支持Http和Https协议，提供了HTTP1.1的部分实现，没有被实现的那部分可以通过扩展已有的Http Server API来实现，程序员必须自己实现HttpHandler接口，HttpServer会调用HttpHandler实现类的回调方法来处理客户端请求，在这里，我们把一个Http请求和它的响应称为一个交换，包装成HttpExchange类，HttpServer负责将HttpExchange传给HttpHandler实现类的回调方法。 六：插入式注解处理API(Pluggable Annotation Processing API) 插入式注解处理API(JSR 269)提供一套标准API来处理Annotations(JSR 175) 实际上JSR 269不仅仅用来处理Annotation，我觉得更强大的功能是它建立了Java 语言本身的一个模型，它把method，package，constructor，type，variable， enum，annotation等Java语言元素映射为Types和Elements(两者有什么区别?)，从而将Java语言的语义映射成为对象，我们可以在javax.lang.model包下面可以看到这些类。 所以我们可以利用JSR 269提供的API来构建一个功能丰富的元编程(metaprogramming)环境。 JSR 269用Annotation Processor在编译期间而不是运行期间处理Annotation，Annotation Processor相当于编译器的一个插件，所以称为插入式注解处理.如果Annotation Processor处理Annotation时(执行process方法)产生了新的Java代码，编译器会再调用一次Annotation Processor，如果第二次处理还有新代码产生，就会接着调用Annotation Processor，直到没有新代码产生为止.每执行一

次process()方法被称为一个"round"，这样整个Annotation processing过程可以看作是一个round的序列。 JSR 269主要被设计成为针对Tools或者容器的API. 举个例子，我们想建立一套基于Annotation的单元测试框架(如TestNG)，在测试类里面用Annotation来标识测试期间需要执行的测试方法。 七：用Console开发控制台程序 JDK1.6中提供了java.io.Console 类专用来访问基于字符的控制台设备。你的程序如果要与Windows下的cmd或者Linux下的Terminal交互，就可以用Console类代劳。但我们不总是能得到可用的Console，一个JVM是否有可用的Console依赖于底层平台和JVM如何被调用。如果JVM是在交互式命令行(比如Windows的cmd)中启动的，并且输入输出没有重定向到另外的地方，那么就可以得到一个可用的Console实例。 八：对脚本语言的支持 如：ruby，groovy，javascript。 九：Common Annotations Common annotations原本是Java EE 5.0(JSR 244)规范的一部分，现在SUN把它的一部分放到了Java SE 6.0中。 随着Annotation元数据功能(JSR 175)加入到Java SE 5.0里面，很多Java 技术(比如EJB，Web Services)都会用Annotation部分代替XML文件来配置运行参数（或者说是支持声明式编程，如EJB的声明式事务），如果这些技术为通用目的都单独定义了自己的Annotations，显然有点重复建设，所以，为其他相关的Java技术定义一套公共的Annotation是有价值的，可以避免重复建设的同时，也保证Java SE和Java EE 各种技术的一致性。 下面列举出Common Annotations 1.0里面的10个Annotations Common Annotations Annotation Retention Target Description Generated Source ANNOTATION_TYPE，CONSTRUCTOR，FIELD

，LOCAL_VARIABLE，METHOD，PACKAGE，PARAMETER，TYPE 用于标注生成的源代码Resource Runtime TYPE，METHOD，FIELD用于标注所依赖的资源，容器据此注入外部资源依赖，有基于字段的注入和基于setter方法的注入两种方式 Resources Runtime TYPE同时标注多个外部依赖，java认证网，加入收藏容器会把所有这些外部依赖注入PostConstruct Runtime METHOD标注当容器注入所有依赖之后运行的方法，用来进行依赖注入后的初始化工作，只有一个方法可以标注为PostConstruct PreDestroy Runtime METHOD当对象实例将要被从容器当中删掉之前，要执行的回调方法要标注为PreDestroy RunAs Runtime TYPE用于标注用什么安全角色来执行被标注类的方法，这个安全角色必须和Container的Security角色一致的。RolesAllowed Runtime TYPE，METHOD用于标注允许执行被标注类或方法的安全角色，这个安全角色必须和Container的Security角色一致的 PermitAll Runtime TYPE，METHOD允许所有角色执行被标注的类或方法DenyAll Runtime TYPE，METHOD不允许任何角色执行被标注的类或方法，表明该类或方法不能在Java EE容器里面运行DeclareRoles Runtime TYPE用来定义可以被应用程序检验的安全角色，通常用isUserInRole来检验安全角色。 注意： 1.RolesAllowed，PermitAll，DenyAll不能同时应用到一个类或方法上 2.标注在方法上的RolesAllowed，PermitAll，DenyAll会覆盖标注在类上的RolesAllowed，PermitAll，DenyAll 3.RunAs，RolesAllowed，PermitAll，DenyAll和DeclareRoles还没有加到Java SE 6.0上来 4. 处理以上Annotations的工作是由Java EE容器来做，Java SE 6.0只是包含了上面表格的前五种Annotations

的定义类，并没有包含处理这些Annotations的引擎，这个工作可以由Pluggable Annotation Processing API(JSR 269)来做。