Java学习路径三篇

# Java学习路径（一）工具篇

# 一、 JDK (Java Development Kit)

　　JDK是整个Java的核心，包括了Java运行环境（Java Runtime Environment），一堆Java工具和Java基础的类库（rt.jar）。不论什么Java应用服务器实质都是内置了某个版本的JDK。因此掌握JDK是学好Java的第一步。最主流的JDK是Sun公司发布的JDK，除了Sun之外，还有很多公司和组织都开发了自己的JDK，例如IBM公司开发的JDK，BEA公司的Jrocket，还有GNU组织开发的JDK等等。其中IBM的JDK包含的JVM（Java Virtual Machine）运行效率要比Sun JDK包含的JVM高出许多。而专门运行在x86平台的Jrocket在服务端运行效率也要比Sun JDK好很多。但不管怎么说，我们还是需要先把Sun JDK掌握好。

## 1、 JDK的下载和安装

　　JDK又叫做J2SE（Java2 SDK Standard Edition），可以从Sun的Java网站上下载到，http://java.sun.com/j2se/downloads.html ，JDK当前最新的版本是J2SDK1.4.2，建议下载该版本的JDK，下载页面在这里：http://java.sun.com/j2se/1.4.2/download.html。

　　下载好的JDK是一个可执行安装程序，默认安装完毕后会在C:\Program Files\Java\目录下安装一套JRE（供浏览器来使用），在C:\j2sdk1.4.2下安装一套JDK（也包括一套JRE）。然后我们需要在环境变量PATH的最前面增加java的路径C:\j2sdk1.4.2\bin。这样JDK就安装好了。

## 2、 JDK的命令工具

　　JDK的最重要命令行工具：

　　java： 启动JVM执行class

　　javac： Java编译器

　　jar： Java打包工具

　　javadoc： Java文档生成器

　　这些命令行必须要非常非常熟悉，对于每个参数都要很精通才行。对于这些命令的学习，JDK Documentation上有详细的文档。

# 二、 JDK Documentation

　　Documentation在JDK的下载页面也有下载连接，建议同时下载Documentation。Documentation是最最重要的编程手册，涵盖了整个Java所有方面的内容的描述。可以这样说，学习Java编程，大部分时间都是花在看这个Documentation上面的。我是随身携带的，写Java代码的时候，随时查看，须臾不离手。

在线Java documentation：

<http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html>

<http://docs.oracle.com/javase/8/docs/>

Java日期和字符串转换：

<http://www.jb51.net/article/95619.htm>l

# 三、 应用服务器(App Server)

　　App Server是运行Java企业组件的平台，构成了应用软件的主要运行环境。当前主流的App Server是BEA公司的Weblogic Server和IBM公司的Websphere以及免费的Jboss，选择其中一个进行学习就可以了，个人推荐Weblogic，因为它的体系结构更加干净，开发和部署更加方便，是Java企业软件开发人员首选的开发平台。下面简要介绍几种常用的App Server：

## 1、 Tomcat

　　Tomcat严格意义上并不是一个真正的App Server，它只是一个可以支持运行Serlvet/JSP的Web容器，不过Tomcat也扩展了一些App Server的功能，如JNDI，数据库连接池，用户事务处理等等。Tomcat被非常广泛的应用在中小规模的Java Web应用中，因此本文做一点下载、安装和配置Tomcat的介绍：

Tomcat是Apache组织下Jakarta项目下的一个子项目，它的主网站是：http://jakarta.apache.org/tomcat/ ，Tomcat最新版本是Tomcat5.0.16，软件下载的连接是：http://www.apache.org/dist/jakarta/tomcat-5/v5.0.16/ 。

　　下载Tomcat既可以直接下载zip包，也可以下载exe安装包（个人建议zip更干净些），不管哪种情况，下载完毕安装好以后（zip直接解压缩就可以了）。需要设置两个环境变量：

　　JAVA\_HOME=C:\j2sdk1.4.2

CATALINA\_HOME=D:\tomcat4 (你的Tomcat安装目录)

这样就安装好了，启动Tomcat运行CATALINA\_HOME\bin\startup.bat，关闭Tomcat运行shutdown.bat脚本。Tomcat启动以后，默认使用8080端口，因此可以用浏览器访问http://localhost:8080来测试Tomcat是否正常启动。

　　Tomcat提供了两个Web界面的管理工具，URL分别是：

　　http://localhost:8080/admin/index.jsp

http://localhost:8080/manager/html

　　在启用这两个管理工具之前，先需要手工配置一下管理员用户和口令。用一个文本工具打开CATALINA\_HOME\conf\tomcat-users.xml这个文件，加入如下几行：

　　<role rolename="manager"/>

　　<role rolename="admin"/>

<user username="robbin" password="12345678" roles="admin,manager,tomcat"/>

　　这样用户“robbin”就具备了超级管理员权限。重新启动Tomcat以后，你就可以使用该用户来登陆如上的两个管理工具，通过Web方式进行Tomcat的配置和管理了。

## 2、 BEA Weblogic

　　Weblogic可以到BEA的网站上免费注册之后下载到Weblogic8.1企业版，License可以免费使用1年时间，其实这已经完全足够了。Weblogic的下载连接：http://commerce.bea.com/index.jsp，Weblogic的在线文档：http://edocs.bea.com/ 。

## 3、 IBM Webshpere

　　Websphere同样可以下载到免费的试用版本，到IBM的developerWorks网站可以看到Websphere试用产品的下载和相关的Websphere的资料，developerWorks中文网站的连接是：http://www-900.ibm.com/developerWorks/cn/wsdd/ ，Websphere的下载连接：http://www7b.software.ibm.com/wsdd/downloads/WASsupport.html 。

## 4、 Jboss

　　Jboss是免费开源的App Server，可以免费的从Jboss网站下载：http://www.jboss.org/index.html，然而Jboss的文档是不免费，需要花钱购买，所以为我们学习Jboss设置了一定的障碍。在Jdon上有几篇不错的Jboss配置文档，可以用来参考：http://www.jdon.com/idea.html

# 四、 Java应用的运行环境

　　Java的应用可以简单分为以下几个方面：

## 1、 Java的桌面应用

　　桌面应用一般仅仅需要JRE的支持就足够了。

## 2、 Java Web应用

　　Java的Web应用至少需要安装JDK和一个web容器（例如Tomcat），以及一个多用户数据库，Web应用至少分为三层：

　　Browser层：浏览器显示用户页面

　　Web层：运行Servlet/JSP

　　DB层：后端数据库，向Java程序提供数据访问服务

## 3、 Java企业级应用

　　企业级应用比较复杂，可以扩展到n层，最简单情况会分为4层：

　　Browser层：浏览器显示用户页面

　　Client层：Java客户端图形程序（或者嵌入式设备的程序）直接和Web层或者EJB层交互

　　Web层：运行Servlet/JSP

　　EJB层：运行EJB，完成业务逻辑运算

　　DB层：后端数据库，向Java程序提供数据访问服务

## 4、 Java嵌入式应用

　　Java嵌入式应用是一个方兴未艾的领域，从事嵌入式开发，需要从Sun下载J2ME开发包，J2ME包含了嵌入式设备专用虚拟机KVM，和普通的JDK中包含的JVM有所不同。另外还需要到特定的嵌入式厂商那里下载模拟器。

# Java学习路径（二）书籍篇

学习一门新的知识，不可能指望只看一本，或者两本书就能够完全掌握。需要有一个循序渐进的阅读过程。我推荐Oreilly出版的Java系列书籍。

　　在这里我只想补充一点看法，很多人学习Java是从《Thinking in Java》这本书入手的，但是我认为这本书是不适合初学者的。我认为正确的使用这本书的方法应该是作为辅助的读物。《Thinking in Java》并不是在完整的介绍Java的整个体系，而是一种跳跃式的写作方法，是一种类似tips的方法来对Java很多知识点进行了深入的分析和解释。

　　对于初学者来说，最好是找一本Java入门的书籍，但是比较完整的循序的介绍Java的语法，面向对象的特性，核心类库等等，在看这本书的同时，可以同步来看《Thinking in Java》，来加深对Java的理解和原理的运用，同时又可以完整的了解Java的整个体系。

　　对于Java的入门书籍，蔡学镛推荐的是Oreilly的《Exploring Java, 2nd Edition》 或者《Java in a Nutshell,2nd Edition（针对C++背景）》，我并没有看过这两本书。其实我觉得电子工业出版社的《Java 2编程详解》或者《Java 2从入门到精通》就很不错。

　　在所有的Java书籍当中，其实最最有用的，并不是O´reilly的 Java Serials，真正最最有用处是JDK的Documentation！几乎你想获得的所有的知识在Documentation里面全部都有，其中最主要的部分当然是Java基础类库的API文档，是按照package来组织的，对于每一个class都有详细的解释，它的继承关系，是否实现了某个接口，通常用在哪些场合，还可以查到它所有的public的属性和方法，每个属性的解释，意义，每个方法的用途，调用的参数，参数的意义，返回值的类型，以及方法可能抛出的异常等等。可以这样来说，所有关于Java编程方面的书籍其实都不过是在用比较通俗易懂的语言，和良好的组织方式来介绍Documentation里面的某个package里面包含的一些类的用法而已。所以万变不离其宗，如果你有足够的能力来直接通过Documentation来学习Java的类库，那么基本上就不需要看其他的书籍了。除此之外，Documentation也是编程必备的手册，我的桌面上有三个Documentation的快捷方式，分别是J2SDK1.4.1的Documentation，Servlet2.3的Documentation和J2SDKEE1.3.1的Documentation。有了这个三个Documentation，什么其他的书籍都不需要了。

对于Java Web 编程来说，最核心的是要熟悉和掌握HTTP协议，这个就和Java无关了，在熟悉HTTP协议之后，就需要熟悉Java的实现HTTP协议的类库，也就是Servlet API，所以最重要的东西就是Servlet API。当然对于初学者而言，直接通过Servlet API来学习Web编程有很大的难度，我推荐O´reilly的《Java Server Pages 》这本书来学习Web 编程。

Servlet API地址：

http://tomcat.apache.org/tomcat-5.5-doc/servletapi/index.html

　EJB的书籍当中，《Enterprise Java Beans?, 2nd Edition》是一本很不错的书， EJB的学习门槛是比较高，入门很难，但是这本书完全降低了学习的难度，特别重要的一点是，EJB的学习需要结合一种App Server的具体实现，所以在学习EJB的同时，必须同步的学习某种App Server，而这本书相关的出了三本书，分别是Weblogic6.1，Websphere4.0和JBoss3.0上面部署书中例子的实做。真是既有理论，又有实践。在学习EJB的同时，可以边看边做，EJB的学习会变得很轻松。

　　但是这本书也有一个问题，就是版本比较旧，主要讲EJB1.1规范和部分EJB2.0的规范。而Ed Roman写的《Mastering EJB 2.0》这本书完全是根据EJB2.0规范写的，深入浅出，覆盖了EJB编程的各个方面，并且还有很多编程经验tips，也是学习EJB非常推荐的书籍之一。

　　如果是结合Weblogic来学习J2EE的话，《J2EE应用与BEA Weblogic Server》绝对是首选读物，虽然是讲述的Weblogic6.0，仍然值得购买，这本书是BEA官方推荐的教材，作者也是BEA公司的工程师。现在中文版已经随处可见了。这本书结合Weblogic介绍了J2EE各个方面的技术在Weblogic平台上的开发和部署，实践指导意义非常强。

　　在掌握了Java平台基础知识和J2EE方面的知识以后，更进一步的是学习如何运用OO的方法进行软件的设计，那么就一定要学习“设计模式”。Sun公司出版了一本《J2EE核心模式》，是每个开发Java企业平台软件的架构师必备的书籍。这本书全面的介绍了J2EE体系架构的各种设计模式，是设计师的必读书籍。

# Java学习路径（三）过程篇

每个人的学习方法是不同的，一个人的方法不见得适合另一个人，我只能是谈自己的学习方法。因为我学习Java是完全自学的，从来没有问过别人，所以学习的过程基本上完全是自己摸索出来的。我也不知道这种方法是否是比较好的方法，只能给大家提供一点参考了。

## 学习Java的第一步是安装好JDK。

写一个Hello World， 其实JDK的学习没有那么简单，关于JDK有两个问题是很容易一直困扰Java程序员的地方：一个是CLASSPATH的问题，其实从原理上来说，是要搞清楚JRE的ClassLoader是如何加载Class的；另一个问题是package和import问题，如何来寻找类的路径问题。把这两个问题摸索清楚了，就扫除了学习Java和使用JDK的最大障碍。推荐看一下王森的《Java深度历险》，对这两个问题进行了深入的探讨。

## 第二步是学习Java的语法。

Java的语法是类C++的，基本上主流的编程语言不是类C，就是类C++的，没有什么新东西，所以语法的学习，大概就是半天的时间足够了。唯一需要注意的是有几个不容易搞清楚的关键字的用法，public，protected，private，static，什么时候用，为什么要用，怎么用，这可能需要有人来指点一下，我当初是完全自己琢磨出来的，花了很久的时间。不过后来我看到《Thinking in Java》这本书上面是讲了这些概念的。

## 第三步是学习Java的面向对象的编程语言的特性。

比如继承，构造器，抽象类，接口，方法的多态，重载，覆盖，Java的异常处理机制。对于一个没有面向对象语言背景的人来说，我觉得这个过程需要花很长很长时间，因为学习Java之前没有C++的经验，只有C的经验，我是大概花了一个月左右吧，才彻底把这些概念都搞清楚，把书上面的例子反复的揣摩，修改，尝试，把那几章内容反复的看过来，看过去，看了不下5遍，才彻底领悟了。不过我想如果有C++经验的话，应该一两天时间足够了。那么在这个过程中，可以多看看《Thinking in Java》这本书，对面向对象的讲解非常透彻。可惜的是我学习的时候，并没有看到这本书，所以自己花了大量的时间，通过自己的尝试和揣摩来学会的。

## 第四步就是开始熟悉Java的类库。

Java的基础类库其实就是JDK安装目录下面jre\lib\rt.jar这个包。学习基础类库就是学习rt.jar。基础类库里面的类非常非常多。据说有3000多个，我没有统计过。

但是真正对于我们来说最核心的只有4个，分别是

　　java.lang.\*;

　　java.io.\*;

　　java.util.\*;

　　java.sql.\*;

　　这四个包的学习，每个包的学习都可以写成一本厚厚的教材，而O´reilly也确实是这样做的。我觉得如果时间比较紧，是不可能通过读四本书来学习。我觉得比较好的学习方法是这样的：

　　首先要通读整个package的框架，了解整个package的class，interface，exception的构成，最好是能够找到介绍整个包框架的文章。这些专门介绍包的书籍的前几章应该就是这些总体的框架内容介绍。

　　对包整体框架的把握并不是要熟悉每个类的用法，记住它有哪些属性，方法。想记也记不住的。而是要知道包有哪些方面的类构成的，这些类的用途是什么，最核心的几个类分别是完成什么功能的。

我在给人培训的时候一般是一次课讲一个包，所以不可能详细的介绍每个类的用法，但是我反复强调，我给你们讲这些包的不是要告诉你们类的方法是怎么调用的，也不要求你们记住类的方法调用，

而是要你们了解，Java给我们提供了哪些类，每个类是用在什么场合，当我遇到问题的时候，我知道哪个类，或者哪几个类的组合可以解决我的问题，That´all！，当我们具体写程序的时候，

只要你知道该用哪个类来完成你的工作就足够了。

编码的时候，具体的方法调用，是边写代码，边查Documentation，所有的东西都在Documentation里面，不要求你一定记住，实际你也记不住3000多个类的总共将近10万个方法调用。

所以对每个包的总体框架的把握就变得极为重要。

## 第五步

通过上面的学习，如果学的比较扎实的话，就打好了Java的基础了，剩下要做的工作是扫清Documentation里面除了上面4个包之外的其他一些比较有用处的类。相信进展到这一步，Java的自学能力已经被培养出来了，可以到了直接学习Documentation的水平了。

除了要做GUI编程之外，JDK里面其他会有用处的包是这些：

　　java.text.\*;

　　java.net.\*;

　　javax.naming.\*;

　　这些包里面真正用的比较多的类其实很少，只有几个，所以不需要花很多时间。

## 第六步Java Web 编程

　　Web编程的核心是HTTP协议，HTTP协议和Java无关，如果不熟悉HTTP协议的话，虽然也可以学好Servlet/JSP编程，但是达不到举一反三，一通百通的境界。所以HTTP协议的学习是必备的。如果熟悉了HTTP协议的话，又有了Java编程的良好的基础，学习Servlet/JSP简直易如反掌，我学习Servlet/JSP就用了不到一周的时间，然后就开始用JSP来做项目了。

　　在Servlet/JSP的学习中，重头仍然是Servlet Documentation。Servlet API最常用的类很少，花比较少的时间就可以掌握了。把这些类都看一遍，多写几个例子试试。Servlet/JSP编程本质就是在反复调用这些类来通过HTTP协议在Web Server和Brower之间交谈。另外对JSP，还需要熟悉几个常用JSP的标记，具体的写法记不住的话，临时查就是了。

　　此外Java Web编程学习的重点要放在Web Application的设计模式上，如何进行业务逻辑的分析，并且进行合理的设计，按照MVC设计模式的要求，运用Servlet和JSP分别完成不同的逻辑层，掌握如何在Servlet和JSP之间进行流程的控制和数据的共享，以及Web Application应该如何配置和部署。

## 第七步J2EE编程

　　以上的学习过程如果是比较顺利的话，进行到这一步，难度又陡然提高。因为上面的知识内容都是只涉及一个方面，而像EJB，JMS，JTA等核心的J2EE规范往往是几种Java技术的综合运用的结晶，所以掌握起来难度比较大。

　　首先一定要学习好JNDI，JNDI是App Server定位服务器资源（EJB组件，Datasouce，JMS）查找方法，如果对JNDI不熟悉的话，EJB，JMS这些东西几乎学不下去。JNDI其实就是javax.naming.\*这个包，运用起来很简单。难点在于服务器资源文件的配置。对于服务器资源文件的配置，就需要看看专门的文档规范了，比如web.xml的写法，ejb-jar.xml的写法等等。针对每种不同的App Server，还有自己的服务资源配置文件，也是需要熟悉的。

　　然后可以学习JTA，主要是要理解JTA对于事务的控制的方法，以及该在什么场合使用JTA。这里可以简单的举个例子，我们知道一般情况可以对于一个数据库连接进行事务控制(conn.setAutoCommit(false),....,conn.commit())，做为一个原子\*作，但是假设我的业务需求是要把对两个不同数据库的\*作做为一个原子\*作，你能做的到吗？这时候只能用JTA了。假设\*作过程是先往A数据库插一条记录，然后删除B数据库另一个记录，我们自己写代码是控制不了把整个\*作做为一个原子\*作的。用JTA的话，由App Server来完成控制。

　　在学习EJB之前要学习对象序列化和RMI，RMI是EJB的基础。接着学习JMS和EJB，对于EJB来说，最关键是要理解EJB是如何通过RMI来实现对远端对象的调用的，以及在什么情况下要用到EJB。

　　在学习完EJB，JMS这些东西之后，你可能会意识到要急不可待学习两个领域的知识，一个是UML，另一个是Design Pattern。Java企业软件的设计非常重视框架(Framework)的设计，一个好的软件框架是软件开发成功的必要条件。在这个时候，应该开始把学习的重点放在设计模式和框架的学习上，通过学习和实际的编程经验来掌握EJB的设计模式和J2EE的核心模式。

　　J2EE规范里面，除了EJB，JMS，JTA，Servlet/JSP，JDBC之外还有很多很多的企业技术，这里不一一进行介绍了。

　　另外还有一个最新领域Web Services。Web Services也完全没有任何新东西，它像是一种黏合剂，可以把不同的服务统一起来提供一个统一的调用接口，作为使用者来说，我只要获得服务提供者给我的WSDL（对服务的描述），就够了，我完全不知道服务器提供者提供的服务究竟是EJB组件，还是.Net组件，还是什么CORBA组件，还是其他的什么实现，我也不需要知道。Web Services最伟大的地方就在于通过统一的服务提供方式和调用方式，实现了整个Internet服务的共享，是一个非常令人激动的技术领域。

Web Services好像目前还没有什么很好的书籍，但是可以通过在网络上面查资料的方式来学习。

0

# Java开发纯文本方式和使用工具的联系与区别

来源：<http://www.iteye.com/problems/90669>

首先了解一下下面几个概念，可能讲得不太准确：   
1、JVM   
JVM是class以及jar(实际上就是很多个class压缩在一起)的运行环境，特征就是java和javaw命令，通过这两个命令，你可以执行class和jar文件。你可以通过-classpath参数指定你需要加载的jar文件   
  
2、JDK   
JDK就是JAVA的命令行开发环境，内置了JVM，特征就是javac命令，这个命令允许你将.java源文件批量或者单个编译成.class文件，从而可以通过JVM的java命令执行。在编译时你可以通过-classpath参数指定你的源代码依赖的jar文件。   
  
3、Tomcat等JEE中间件   
JEE中间件主要是为了让JAVA程序能够提供http服务、向客户展现html及相关资源而准备的一个运行环境，通常已经包含了JDK（或者像tomcat一样需要配置JDK所在路径）.这个运行环境的特征是能够让你部署一个war包，运行环境能够自动加载WEB-INF/classes下的.class文件和WEB-INF/lib下jar文件。当用户通过浏览器访问中间件中你的war包所部署的路径时，中间件能够按照J2EE标准调用你的war包中的class和jsp页面，并将执行结果返回给浏览器。 在这种情况下你只需要将你的.java文件编译好放到WEB-INF/classes目录下。   
  
然后说说纯文本JAVA WEB开发，有这么几种类型：   
1、纯文本只有JSP   
如果项目中只有jsp页面，就很稀松平常了，你只需要在tomcat的webapps下新建一个目录，然后在里面新建.jsp文件，就可以通过类似于http://localhost:8080/app/a.jsp的方式执行这个jsp文件了。早期的JAVA WEB项目都差不多是这么干的，那时候IDE的用处确实不大。别的技术像ASP和PHP之类十几年了一直是用这种纯文本的方式来编写代码，效率和有IDE没太大差别。   
  
2、除了JSP还有辅助JAVA类   
如果除了JSP以外，你还需要一些JAVA类来辅助JSP页面，这时候你必须手工新建.java文件，然后用通过文本编辑器（记事本/vi/UE等）打开他，往里面手工写代码。你将代码写完后，还需要将这个.java文件编译成.class，然后才能放到WEB-INF/classes下执行，   
有两种方式完成编译动作：   
方式一：JDK+构建工具（ANT或者MAVEN），通过配置ANT调用JDK中的javac命令将你的JAVA代码编译成class并放置到你的WEB-INF/classes目录下   
方式二：ANT实际上也是将你的配置转换成javac命令中的各种参数，所以你也可以直接在命令行输出javac，然后javac就会提示你需要提供什么样的参数以及各种参数有什么用途，你可以根据提示自己调用javac编译.java文件成.class文件。   
两种方式效率都有点低，所以你还需要有一个.bat（或者.sh）批处理文件来帮助你每隔几十秒就自动编译一下指定目录下的.java文件。   
你可能需要写的辅助JAVA类有：   
一、全局性的ServletFilter（例如用于用户权限检查）   
二、复杂输出的Servlet(jsp不适合用来动态向浏览器提供图片等资源，这时候写Servlet)   
三、在JSP中使用的Tag类，以减少JSP中大量的<%%>代码。   
这种类不会很多，一般项目能有二三十个就差不多了，所以最初写辅助类的痛苦时期过去以后，开发效率就和使用IDE没什么区别了   
  
3、除了JSP还有大量JAVA类   
你的系统比较庞大了，需要一些框架性的东西来进行总体约束，然后在按照框架的要求来编写大量的JAVA类，通过这些JAVA类来完成浏览器端的请求。   
这时候你的系统状态已经比较类型与SSH这种形态了，每个页面都会有对应的Action/Controller，以及FormBean/Model，或者还有专门的Service/BL类和DAO类。   
这样你的系统中就会有大量的.java文件需要编译成.class，而且要引用的jar数量也显著增加，这时候你作为一个没有IDE的人，痛苦就要来了：   
大量的时间花在等待编译上了，无论是使用构建工具的增量编译功能还是自己写一个更强大的批处理文件来扫描改动过的类，每次编译都需要一点时间，并且都不是非常可靠。特别是“减量编译”通常都支持不好，可能会有潜在的错误。   
如果每次都使用全部重新编译以得到可靠的编译结果，那么最好的做法是让你每个JAVA类都一次写完一次编译成功，如果多出几个错误或者多修改几次，你会发现编译用的时间就显著大于你写代码的时间。   
这种情况下纯文本和IDE的开发效率会有显著的区别。   
如果我必须在这种痛苦的纯文本条件下工作，我会先写一个JAVA类，让这个JAVA类单独开一个进程运行，这个进程会自动扫描相应目录下的java文件增/改/删,   
然后调用sun.tools.javac.Main类来进行编译，如果只是新增和修改则只单独编译几个类，如果有删除则全部重新编译。   
  
那么Eclipse做了些什么？   
1、Eclipse提供了WEB项目工程向导，帮助你快速创建项目   
2、Eclipse将你的WEB-INF/lib下的jar文件管理起来，并对其中的class类进行了索引，以便于你快速查看相关类的位置、继承关系、引用关系。   
3、Eclipse提供一个完善的增量编译器，所写即所得。Eclipse当年能够胜出的一个重要原因就在他的编译器，Eclipse的JDT实现了自己的编译器（因此Eclipse甚至都不需要JDK，只要有JVM就可以了）,能够快速、增量地将你对代码的修改反映到class文件上。这是当时的JBuilder望尘莫及的，当时每次改完类都得重新编译一下，那个痛苦呀，当时我在公司引进Eclipse的时候，两个月之内大家就全部抛弃JBuilder X了（当时花了钱买了正版）。   
4、提供了完善的调试功能，基于IDE的调试效率会远高于命令行调试。   
5、提供了语法加亮、语法提示、中间件管理等辅助工具。

Eclipse是一个为开发者提供一个界面友好，方便快捷的开发工具，本身并不参与实际的程序运行，加载JAR包等都是JVM做的，跟Eclipse没关系。