**3、具体内容**

Eclipse本身是一个工具，但是对于开发工具，个人认为：程序代码会写了，再去使工具。

**3.1、Eclipse历史**

Eclipse中文翻译为日蚀，指的是吞没一切的光芒，那么这个极其具备挑衅意味的名字实际上是针对于SUN公司。在2000年之后一直处于互联网低潮，而SUN公司自从这一低潮之后就再也没有起来过，而后SUN公司最为尴尬的事，他在硬件上没有过多的收益，而且软件的编程语言Java也不是SUN的赚钱工具，它只是变成了一个出卖版权的公司了。

对于Java的IDE（集成开发环境）本身也是经过了一些历史的调整：

· 1995年的时候，Java诞生，但是SUN公司却高调宣布，我们自己不生成IDE，给其他第三方公司生产；

· 在Java产生之后，Borland公司开始入手Java的IDE生产，所以后来的JBuilder就成为了Java开发的唯一选择（不选择JBuilder都使用记事本，IDEA、JCREATOR、EDITPLUS…），后来一直到了JBuilder X之后才正常；

· 2003年之后有了Eclipse，在2004年之后有了SUN自己的开发工具 —— NetBeans。

而从整个国内的Java开发模式上也一直在发生着改变：

· 豪华级架构：操作系统 + 数据库 + 中间件 + 开发工具；

|- IBM体系：AIX + IBM DB2 + Websphere Application Server + WSAD；

|- 超级杂牌军体系：UNIX + Oracle + BEA WebLogic + JBuilder；

· 免费架构：Linux + MySQL + Tomcat / JBoss + Eclipse；

而对于开发工具最早的霸主就是JBuilder，但是遗憾的是，在2006年的时候Borland倒闭了，倒闭的原因是因为没有赶上所谓的开源风潮，因为有许多的学习者，他可能并不需要如此专业的工具，所以当时Borland公司的市场就出现了问题，于是倒闭就成为了显示，最后Borland技术部单独成立了一家技术公司，继续从事于JBuilder的研究，但是今天的JBuilder已经基本上无法去问津了，完全被Eclipse取代了。

Eclipse是由IBM开发的，之后将其转送给了今天的Eclipse组织，进行开源项目的推广，而Eclipse的前身是IBM Visual Age，而后IBM根据Eclipse的1.0模型，产生后来的WSAD开发工具，Eclipse本身是免费的，但是其是靠插件收费（中国连插件都不需要收费），其本身有以下几个基本组成：JDT、JUNIT测试工具、CVS客户端、插件开发。用户可以直接登陆“**www.eclipse.org**”下载Eclipse的最新版，JUNO（朱诺，小行星的名字）版，而且Eclipse现在可以直接解压缩后使用，不再需要单独啊安装，是一个纯粹的绿色版。

分析原因：可能是由于电脑上本身存在了Oracle，所以Eclipse找不到正确该使用的JDK 1.7，那么现在可以在环境属性之中增加一个JAVA\_HOME环境属性，让其定义在JDK的安装目录。

此时很郁闷的是，Eclipse有一个bug，也许以后可以解决。在eclipse的安装目录下有一个eclipse.ini的文件，修改一下，把256修改为128。

|  |
| --- |
| -startup  plugins/org.eclipse.equinox.launcher\_ 1.3.0 .v20120522-1813.jar  --launcher.library  plugins/org.eclipse.equinox.launcher.win32.win32.x86\_1.1.200.v20120522-1813  -product  org.eclipse.epp.package.java.product  --launcher.defaultAction  openFile  --launcher.XXMaxPermSize  **128M**  -showsplash  org.eclipse.platform  --launcher.XXMaxPermSize  **128m**  --launcher.defaultAction  openFile  -vmargs  -Dosgi.requiredJavaVersion=1.5  -Dhelp.lucene.tokenizer=standard  -Xms 40m  -Xmx 512m |

Eclipse启动之后，会出现一个对话框，这个对话框的主要目的是询问是否建立一个项目的工作区，在Eclipse之中，每一个工作区之中可以包含多个Java项目。

现在将工作区建立在d:\myeclipsews目录之中。

**3.2、JDT使用**

启动了Eclipse之后，下面建立一个新的Java项目：MyJavaProject。

之后输入的项目名称：MyJavaProject。

当Java项目建立完成之后，可以在项目的目录下发现两个文件夹：

· src：是保存所有的\*.java程序，都是按照包名称进行保存的；

· bin：保存所有生成的\*.class文件，按照包名称进行保存的。

下面建立一个新的类：TestDemo.java。

使用Eclipse本身最大的好处是在于方便的进行随笔提示，并且在每一次\*.java程序保存的时候，会由Eclipse自动的将其编译为\*.class文件。

另外在Eclipse之中还可以存在一些代码的生成工具，例如：现在有如下一个简单的类：

|  |
| --- |
| **package** cn.mldn.vo;  **public** **class** Person {  **private** String name ;  **private** **int** age ;  } |

这个类肯定是作为简单Java类出现，那么现在很明显简单Java类的开发原则：

· 所有属性封装，已经封装；

· 生成setter、getter方法，〖Source〗 🡺 〖生成getter和setter方法〗

· 构造方法，必须要有无参构造，〖Source〗 🡺 生成构造方法。

· 覆写Object类之中的一些操作方法，〖Source〗 🡺 〖覆写或实现方法〗。

或则使用JUNO版的新功能，自动生成toString()；

不过，除了以上的生成方式，还可以通过快捷键生成代码。

· ALT + /：代码自动补充提示；

· CTRL + 1：进行错误代码纠正提示；

· CTRL + D：删除当前行代码；

· CTRL + SHIFT + O：组织导入所需要的包.类；

· CTRL + SHIFT + F：格式化代码显示；

· CTRL + /：注释或者取消注释当前行代码；

· CTRL + SHIFT + L：所有快捷键列表；

· CTRL + H：强搜索代码；

项目也可以进行删除操作，但是在删除项目的时候有两种方式：

· 方式一：是从项目的工作区之中删除项目，以后可以恢复；

· 方式二：彻底从硬盘上删除项目，彻底删除。

如果要想导入一个项目，则可以使用导入的方式完成。

之后选择项目文件所在的路径。

以上是通过导入的方式完成了项目的导入，而现在也可以利用导出的方式，将一个项目之中的所有\*.class文件自动生成\*.jar文件。

但是如果一个项目之中，需要使用其他的jar包的话，那么不能依靠之前的classpath配置，需要在Eclipse之中单独配置jar包，找到项目属性的构建路径完成。

在Eclipse之中，为了方便用户的开发，还提供了DEBUG功能，可以利用此功能进行项目的调试操作，而如果要想进行调试，那么首先需要设置断点：断点指的是程序执行到此处的时候，自动停止，而后交由人工控制执行。

**范例：**定义要调试的程序

|  |
| --- |
| **package** cn.mldn.demo;  **class** TestMath {  **public** **static** **int** div(**int** x, **int** y) **throws** Exception {  **int** result = 0;  result = x / y;  **return** result;  }  }  **public** **class** TestDemo {  **public** **static** **void** main(String[] args) **throws** Exception {  **int** tempX = 10;  **int** tempY = 2;  **int** res = TestMath.*div*(tempX, tempY); // 在此行的左边空栏上双击鼠标左键  System.*out*.println("计算结果：" + res);  }  } |

这个时候会自动的出现一个断点（Blue Point），而后以Debug的方式启动程序。

随后会出现一个信息提示，询问：是否要切换到调试视图上，选择“yes”，进入到调试视图。

进入到调试视图之后，可以通过以下的四个方式进行代码调试：

· 单步跳入（F5）：进入到代码之中，观察代码的执行；

· 单步跳过（F6）：不关心代码之中的执行，只关心最终的结果；

· 单步返回（F7）：返回到单步跳过的状态；

· 恢复执行（F8）：不再调试，直接将程序运行完成。

在每一个Java程序运行的时候都可以设置一些初始化的配置参数，Eclipse也可以，但是麻烦一些，例如：下面建立一个新的类：PrintParam

|  |
| --- |
| **package** cn.mldn.demo;  **public** **class** PrintParam {  **public** **static** **void** main(String[] args) {  **for** (**int** x = 0; x < args.length; x++) {  System.*out*.println(args[x]);  }  }  } |

如果要想配置初始化参数，则进入到运行时配置，可是如果要想真正配置，必须保证要配置的程序先执行一次。

**3.3、JUNIT测试工具**

软件测试行业：2003年 ~ 2006年之间，各个公司一直都在招聘软件测试人员，而且都是外包到IBM或微软的公司进行测试。但是，看今天的软件测试来讲，这个行业已经很难了。

软件测试是一个黄金职位，对于软件测试是需要有一套完整测试理论和数据设计的，而如果学习过软件工程应该知道软件测试分为两类：

· 黑盒测试：主要是测试功能，例如一些xx版，是不接触代码的，但是公司会提供测试工具；

· 白盒测试：指的性能测试，或者是算法调整。

而这几年随着行业的发展，实际上又给出了一种新的测试职位 —— **Use Case（用例）测试工程师**，对于这种职位而言，国内的平均待遇是程序员的3~5倍，如果按照正常的一个程序员的工资是8000来算，那么这种人的平均月薪就是：2 ~ 5W，但是人很难找。一般而言，此类人员在行业之中需要具备8 ~ 10年的项目经验，并且精通业务。

一般而言，年龄在28岁以下的人员，找工作都是凭着技术，而业务是随着项目走的，28岁 ~ 35岁之间，干代码是可以干，但是基本上拼不起了，30岁之后你靠的行业业务和管理能力。

JUNIT是一个比较常用的测试工具，专门可以进行Use Case测试的，在日后所开发的程序里面都要写大量的JUNIT测试程序。

**范例：**定义要测试的程序

|  |
| --- |
| **package** cn.mldn.util;  **public** **class** MyMath {  **public** **static** **int** add(**int** x, **int** y) {  **return** x + y;  }  **public** **static** **int** sub(**int** x, **int** y) {  **return** x - y;  }  **public** **static** **int** mul(**int** x, **int** y) {  **return** x \* y;  }  **public** **static** **int** div(**int** x, **int** y) {  **return** x / y;  }  } |

下面如果要想测试的话，则需要建立一个新的JUNIT测试端。

建立JUNIT测试的时候有两种形式：

· JUNIT Test Case：表示一个测试用例，主要完成一个业务的测试；

· JUNIT Test Suite：一组测试用例，包含了多个Test Case。

由于JUNIT属于第三方的程序包，那么在使用的时候需要配置相关的\*.jar文件，此处会提示用户是否需要自动进行配置，选择OK即可。

**范例：**编写测试程序

|  |
| --- |
| **package** cn.mldn.util;  **import** junit.framework.TestCase;  **import** org.junit.Test;  **public** **class** MyMathTest {  @Test  **public** **void** testAdd() {  TestCase.*assertEquals*(MyMath.*add*(10, 20), 30);  }  @Test  **public** **void** testSub() {  TestCase.*assertEquals*(MyMath.*sub*(20, 10), 10);  }  @Test  **public** **void** testMul() {  TestCase.*assertEquals*(MyMath.*mul*(20, 10), 200);  }  @Test  **public** **void** testDiv() {  TestCase.*assertEquals*(MyMath.*div*(20, 10), 2);  }  } |

对于JUNIT的测试结果来讲，一共分为两类：

· GREEN BAR：测试通过；

· RED BAR：测试失败。

在日后的所有开发之中，一定要进行程序的分层，其中业务层的测试工具就通过JUNIT完成，包括日后学习课程之中，也一样要编写测试用例。

工具本身不麻烦，难就难在，如何编写测试用例，数据如何设计？

项目经理只管理三个原则：时间、成本、利润。

正常思路：付先生的一生

· 21 ~ 22岁大学毕业：小工（开发团队：5个人，PM，老大，带着几个小工）；

· 23 ~ 25岁之后：如果在一个公司时间长了，慢慢成为主力了，当组长，半个老大，带领新人开发；

· 25 ~ 26岁之后：如果有一定的管理能力、有一定的文档能力，当个PM；

· 26 ~ 30岁之间，积累业务能力，同时阔宽人脉；

· 30岁之后靠你的业务能力存活。

**4、总结**

1、 Eclipse傻子才不会；

2、 JUNIT不是关键，关键是业务。

**5、预习任务**

JDK 1.5的新特性、多线程。

**6、作业**

1、 把六个开发模型都熟悉一遍；

2、 总结问题。