# 1、课程名称：Eclipse 开发工具


# 2、知识点

## 2.1、上次课程的主要知识点

1、 线程的两种实现方式及区别

## 2.2、本次预计讲解的知识点

1、 认识一下 Eclipse 的主要作用
2、 JDT 的使用
3、 JUnit 测试工具的使用
4、 CVS 服务的使用
5、 关于 Java 中的 Annotation 的作用

# 3、具体内容

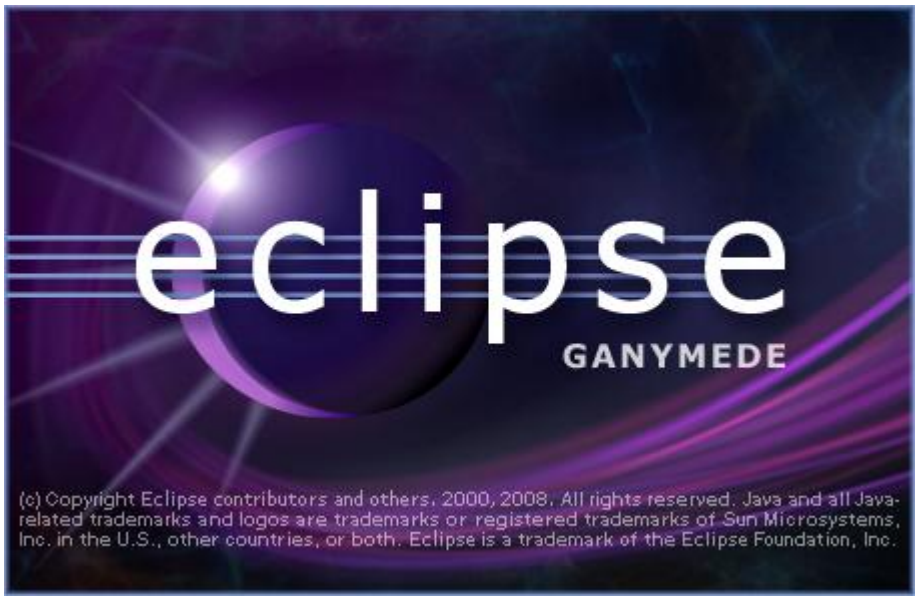
## 3.1、认识 Eclipse（了解）

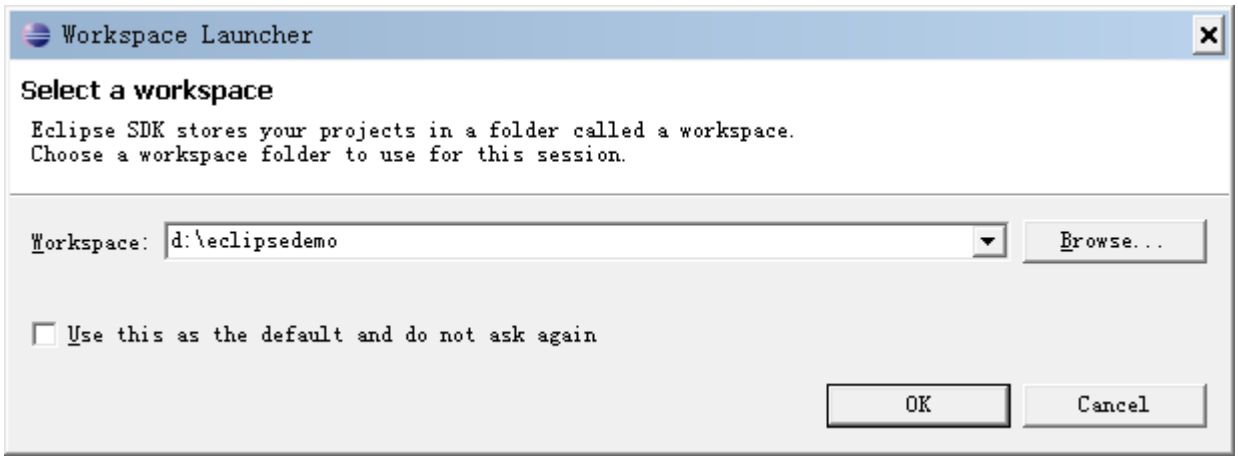
Eclipse 中文的意思表示的是日蚀，表示的是吞没一切的光芒（SUN），最早是由 IBM 开发的，其前身是 **Visual Age**，之后由 IBM 将其转送给了现在的 Eclipse 组织，从而作为开源项目进行推广。

Eclipse 虽然其本身的开发平台是免费的，但是里面的插件是收费的，所以各个公司都开发自己的插件，进行收费的操作，Eclipse 在应用中主要有以下几个组件：

- JDT：Java 的设计开发工具
- Junit 测试端，直接进行测试程序的编写
- CVS 客户端：可以直接与服务器进行连接，从而进行多人开发
- 插件开发：开发一个个可以使用的组件程序。

Eclipse 现在的版本很多，咱们此次使用的是 Eclipse 3.4 版本的开发包，这些软件都属于绿色软件，可以直接解压缩后使用。

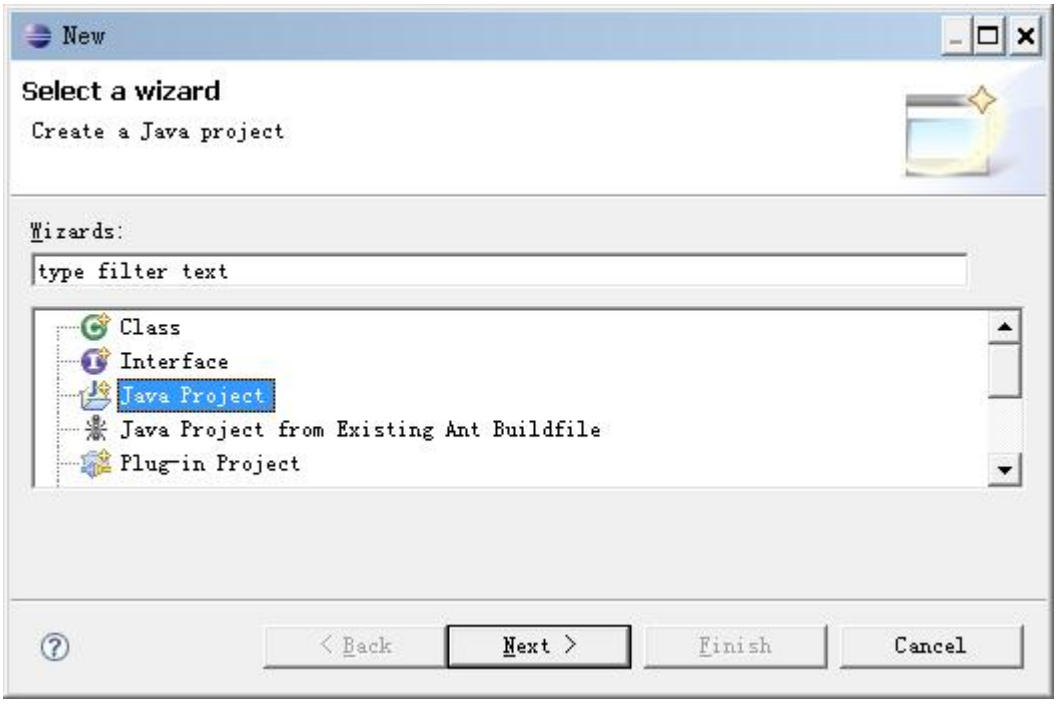


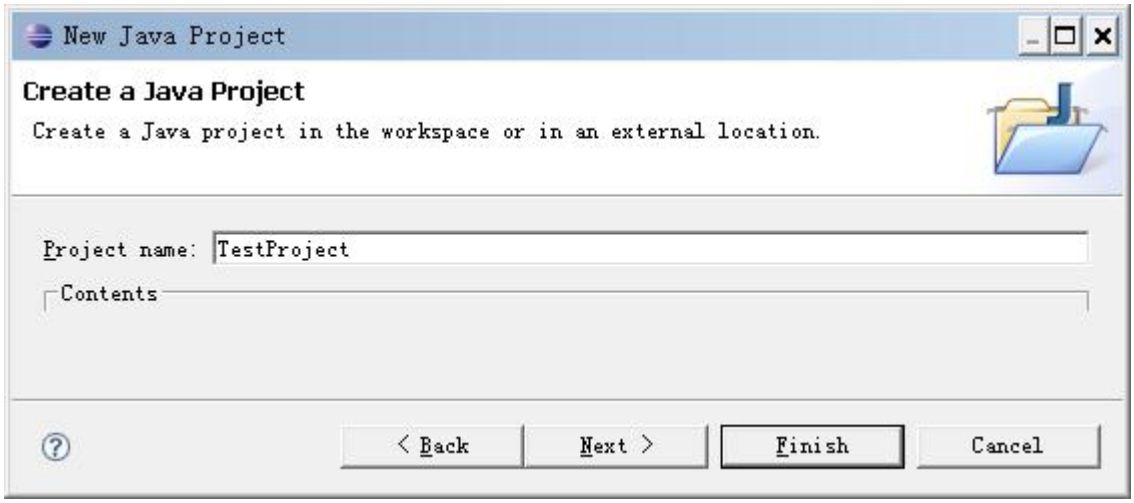




在一个工作区中可以同时建立多个项目。

# 3.2、使用 JDT（重点）

JDT 是一个开发 Java 程序的基本环境，可以直接在里面建立 Java 的项目。




项目名称为：TestProject。

之所以使用 Eclipse 进行开发，最大的好处是在于 Eclipse 可以在每次保存*.java 文件的时候自动为其编译成*.class 文件，而避免掉之前手工编译的麻烦。

在使用 Eclipse 建立类的时候一定要把握住一个原则，即：没有包的类是不存在的，而且要注意命名规范：

```
package org.lxh.demo;
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!!");
    }
}
```


由于其在每次保存的时候都可以自动编译成*.class 文件，所以，直接运行即可。

而且在 Eclipse 中最大的好处还在于提供了以下的功能：

1、 随笔提示

2、 自动的语法检查

3、 代码的修改建议：CTRL + 1

```
package org.lxh.demo;
class MyMath {
    public int div(int x, int y) throws Exception {
        return x / y;
    }
}
public class Hello {
    public static void main(String[] args) {
        MyMath my = new MyMath();
        int temp = 0;
```

```
        try {
            temp = my.div(10, 2);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("计算结果：" + temp);
    }
}
```

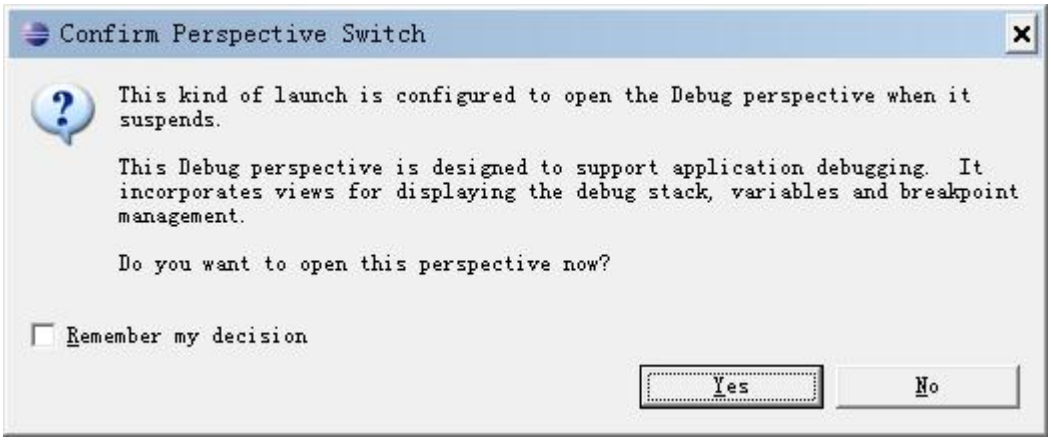
在 Eclipse 开发工具还有代码的生成功能，例如：自动生成 getter 和 setter。

我知道的快捷键：

- 格式化：CTRL + SHIFT + F
- 导入所需要的包.类：CTRL + SHIFT + O
- 复制当前行：CTRL + ALT + ↓

在 Eclipse 中还存在着 DEBUG（调试）的功能，要想使用此功能则肯定需要首先设置断点（Break Point）。

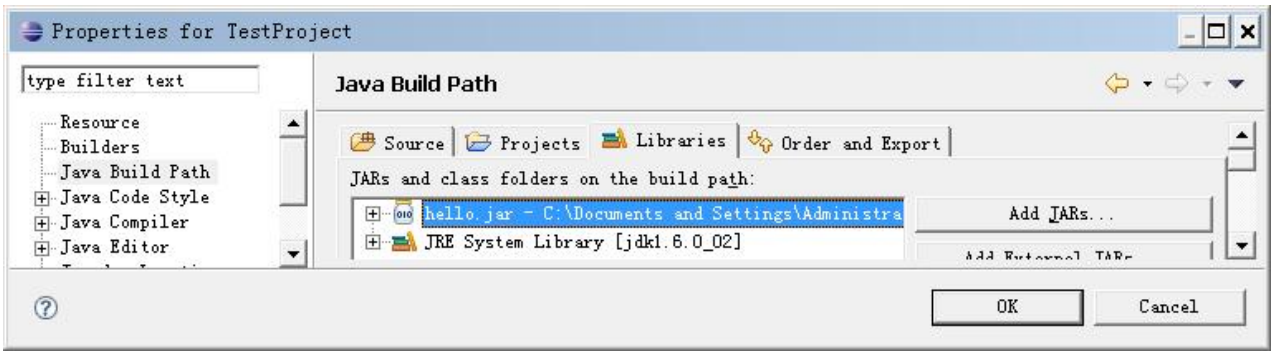
- 在需要停止的代码处，双击左键增加断点。
- 使用 DEBUG 的方式运行。
- 此时出现是否进入到 DEBUG 界面的提示窗口。



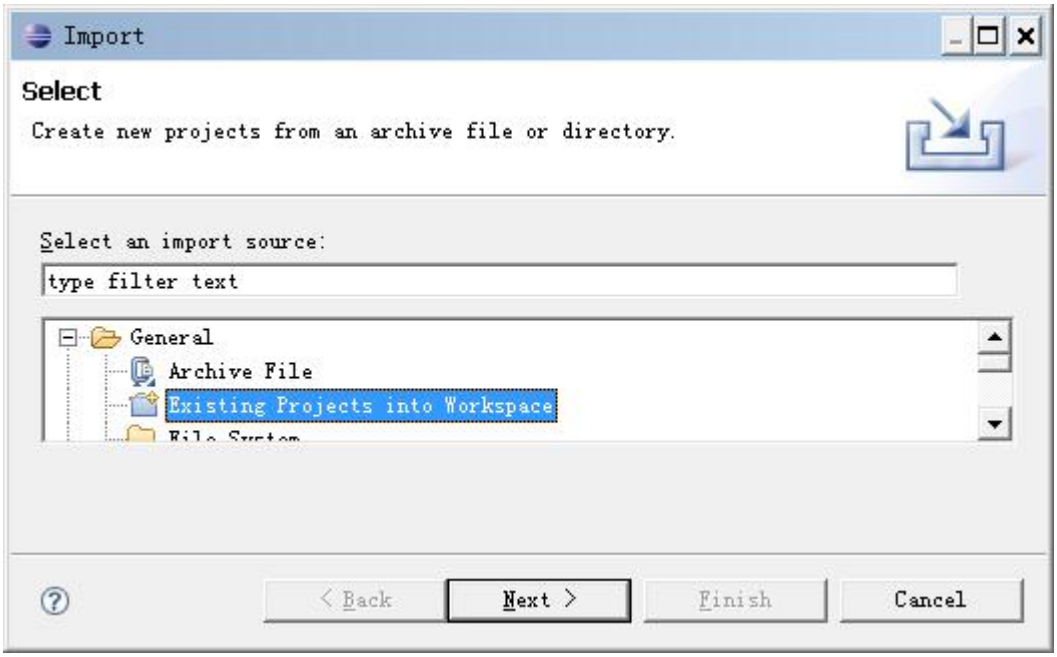

- 进入到 debug 视图之后，可以通过以下的命令进行操作：
  - |- 单步跳过【F6】：在本代码中执行
  - |- 单步跳入【F5】：进到指定操作的内部观察执行过程
  - |- 恢复执行【F8】：代码正常执行完毕

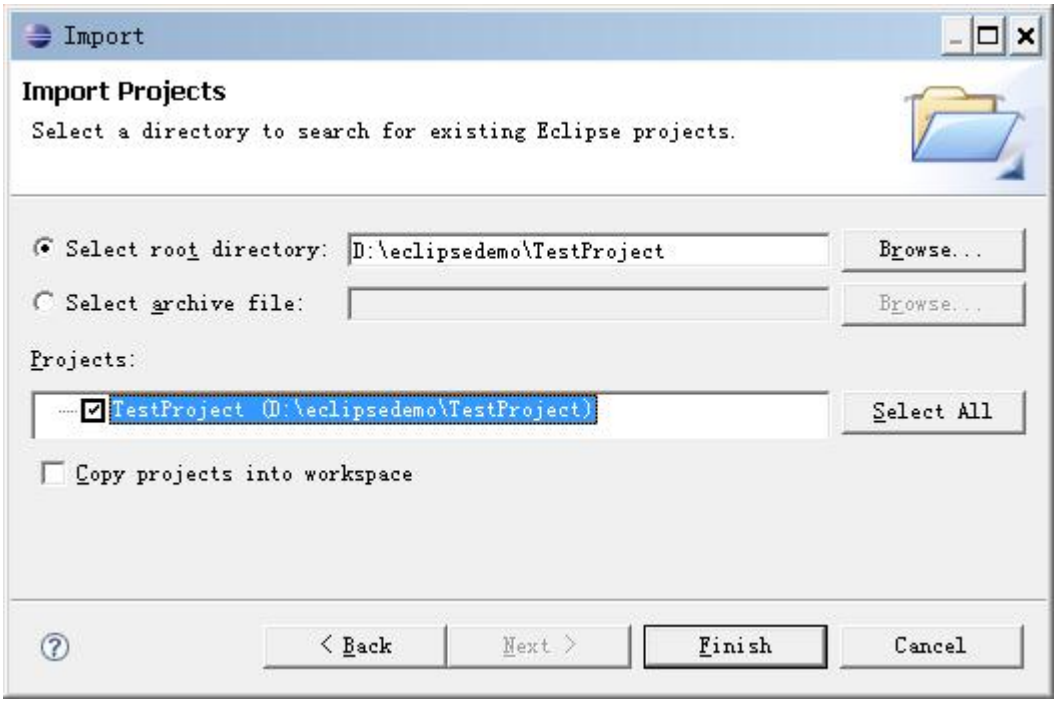
还可以通过导出的命令，将类变为 jar 文件的形式保存：File à Export à Java à JAR File

如果现在在程序中需要其他的 JAR 包操作的话，则可以按照如下步骤进行：项目 à 属性 à 添加 JAR 包




导入工作区：




## 3.3、JUnit 测试工具（理解）

在软件测试中有两种测试方式：

- 黑盒测试：测试功能
- 白盒测试：测试性能

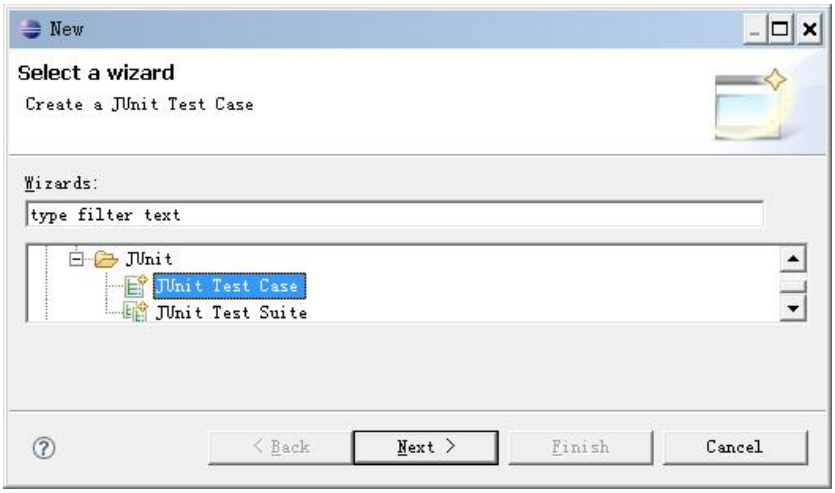
以后如果要想继续做测试的话，最好是做白盒测试，编写 USE CASE（测试用例），但是要想达到这一步必须清楚的掌握一个行业的完整的业务流程。

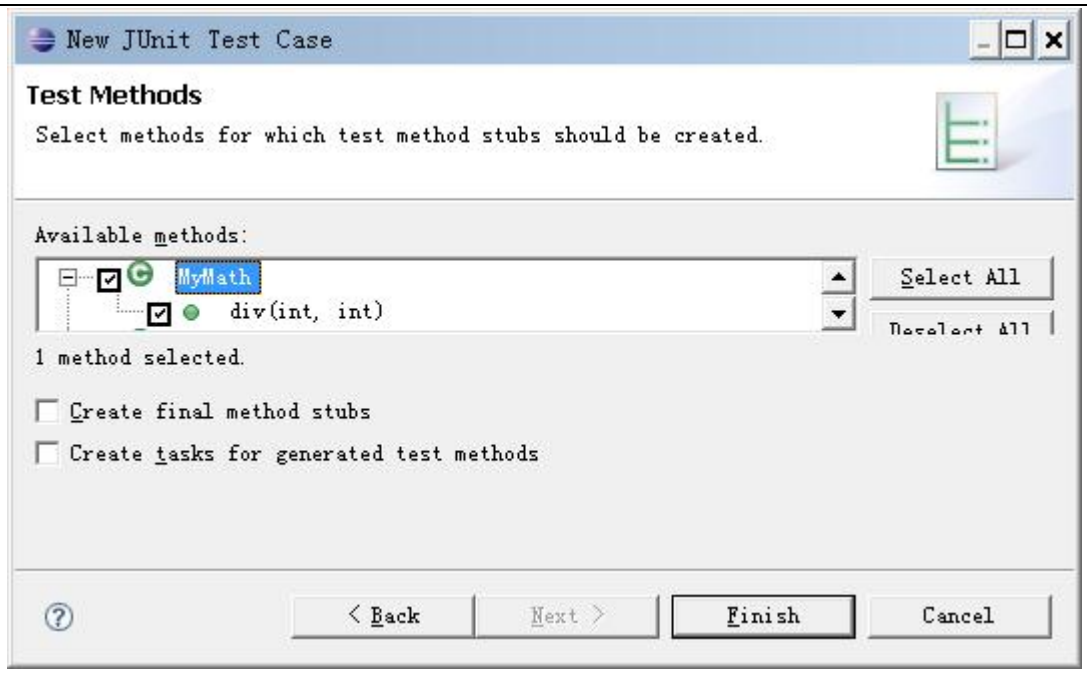
在进行白盒测试的时候可以使用的工具有很多，但是 Junit 使用算是一个最广泛的，最早的时候此工具是以一个单独的组件包的形式出现的，需要进行配置，但是在 Eclipse 之后已经将此组件成功的加入到了开发工具之中，所以直接建立即可。

**范例：**有如下的程序需要测试

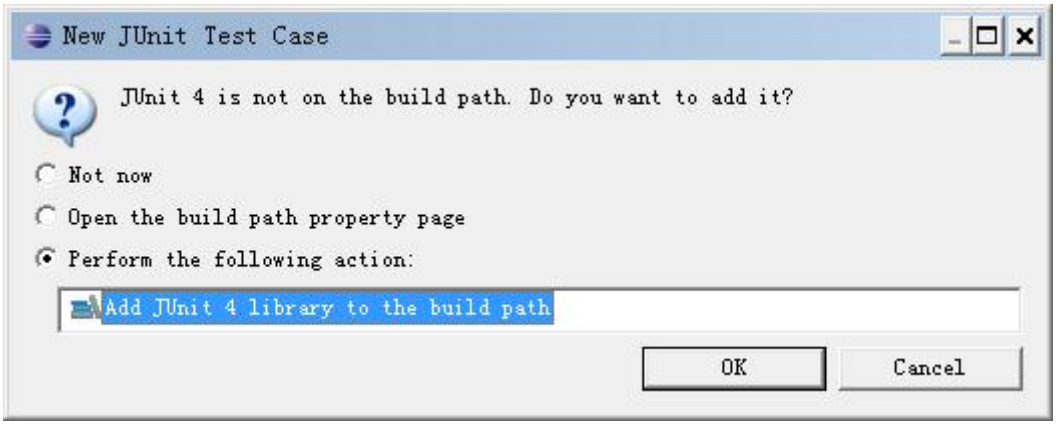
```
package org.lxh;
public class MyMath {
    public int div(int x, int y) throws Exception {
        int temp = 0;
        temp = x / y;
        return temp;
    }
}
```

之后建立一个 Junit 的测试用例，多个测试用例称为一个测试站点。




现在要对 MyMath 类中的 div()方法进行测试。




Junit 本身是一组 JAR 包，所以要使用的话一定要在 CLASSPATH 环境中进行配置，此处询问的是否将 jar 包加入到项目之中。

```
package org.lxh;
import junit.framework.Assert;
import org.junit.Test;
public class MyMathTest {
    @Test
    public void testDiv() {
        try {
            Assert.assertEquals(new MyMath().div(10, 2), 5);
        } catch (Exception e) {
        }
    }
}
```

本程序的功能就是测试 10/2 的内容是否是 5，如果是 5，则通过，如果不是，则不通过。

那么对于测试的结果有两种返回内容：

- GREEN BAR：表示测试成功
- RED BAR：表示测试失败

工具本身使用并不麻烦，难的就在于如何去编写一个测试用例。

如果要想做到这一步，需要补测试的原理方面的内容。

## 3.4、CVS 服务器（理解）

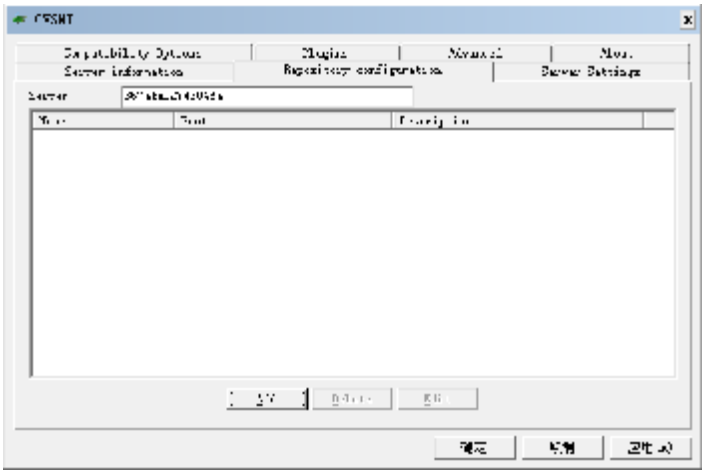
在进行多人开发的时候，往往所有的开发者都需要向服务器上提交自己的代码，但是对于这些代码就必须进行严格的管理，例如：某一个人正在修改的时候其他人不应该修改，这样的功能就可以通过版本控制器完成，其中 CVS 就是一个比较著名的版本控制器，大但是现在随着发展已经不再使用，现在最主流的版本控制工具是 SVN。

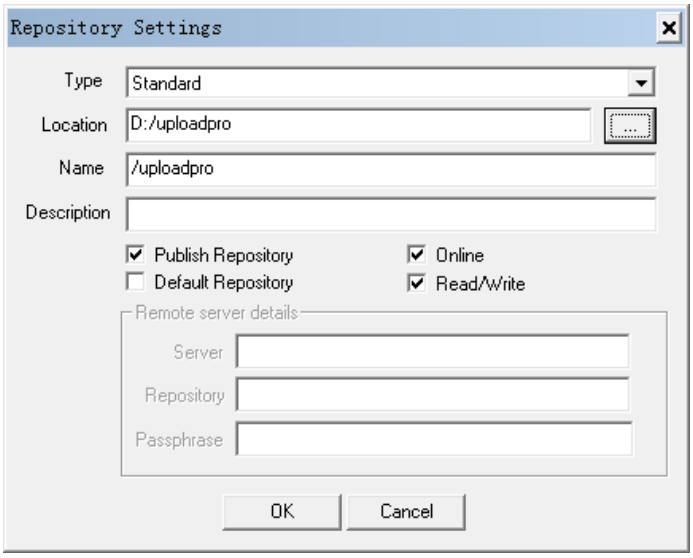
CVS 本身需要服务器端和客户端才可以使用，其中客户端在 Eclipse 中已经集成了，下面分别来看如何使用。

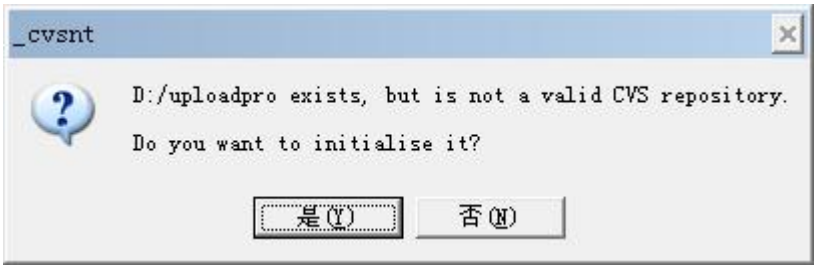
### 3.4.1、搭建 CVS 服务器端

几乎所有的版本控制器都是从 Linux 下发展而来的，CVS 本身也一样，在 windows 中如果要想使用 CVS，则需要先安装 CVSNT 的服务器端。

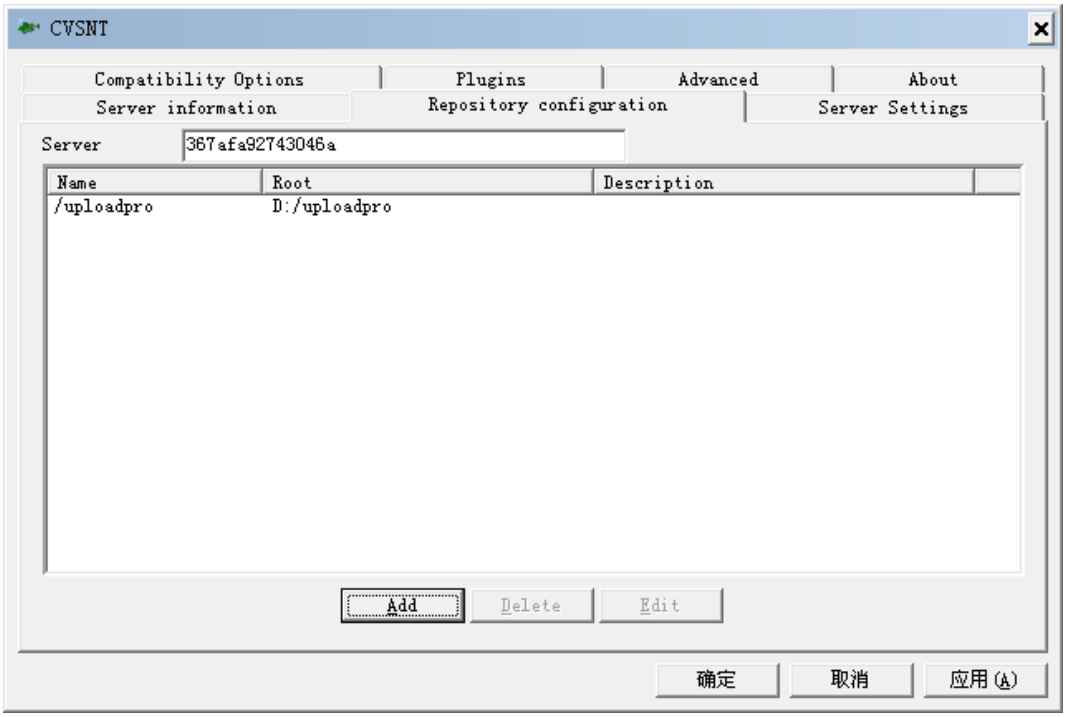
CVS 服务器安装完成之后就需要进行配置，例如：现在需要将所有的代码都保存在 d:\uploadpro 目录之中。

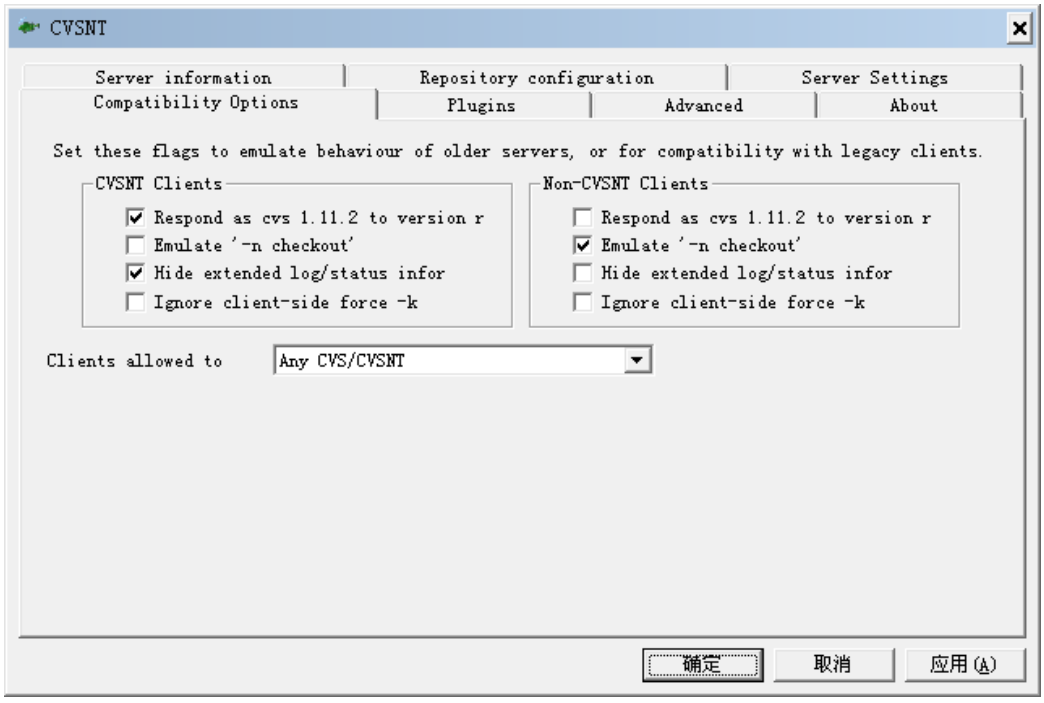







一旦配置好了一个目录之后，需要向此目录中增加配置文件，此处选择“是”。

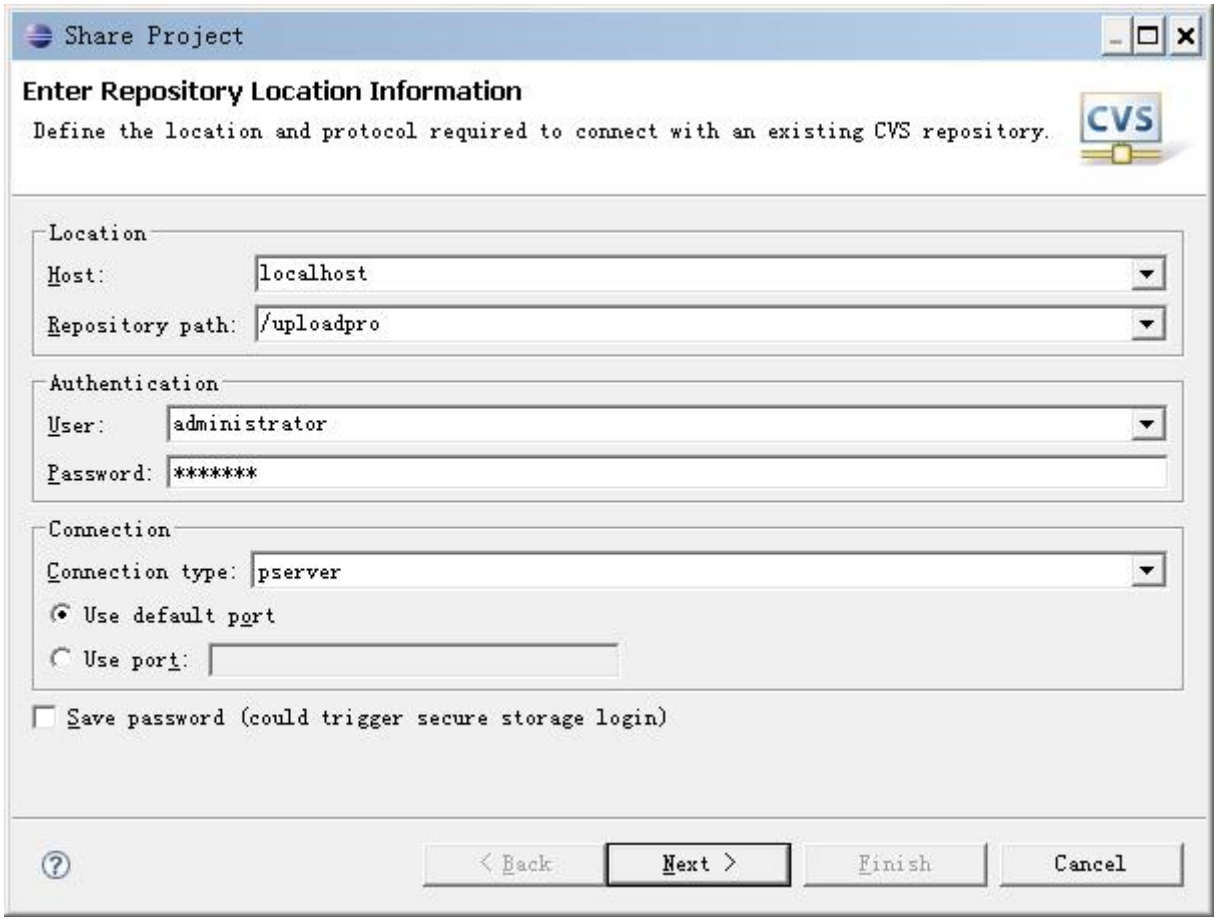


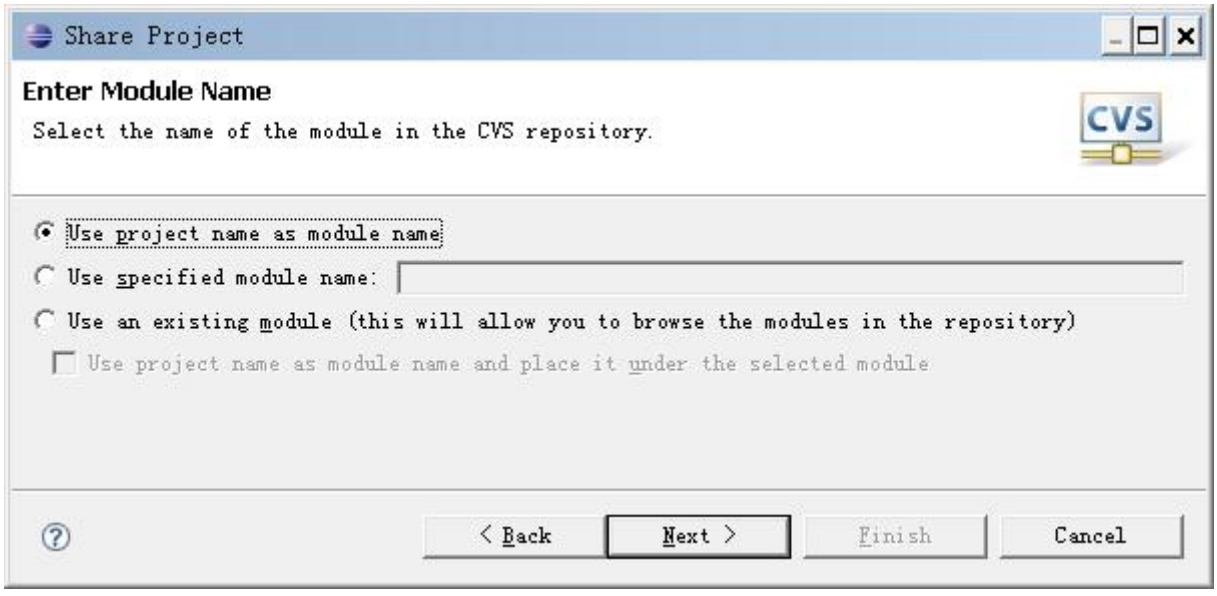



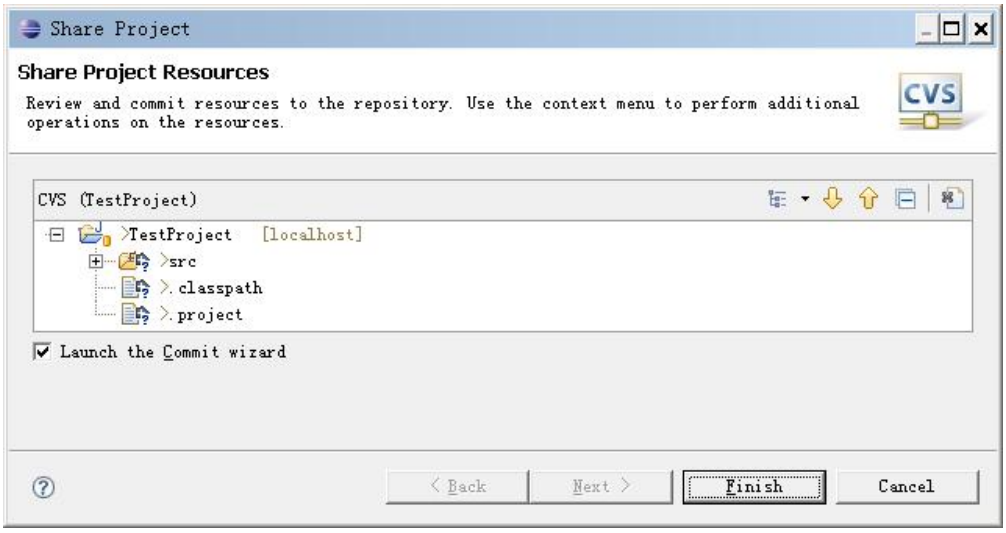
# 3.4.2、搭建 CVS 客户端

Eclipse 中直接集成了客户端，所以直接配置即可。

项目 à Team à Share Project à 输入各种信息

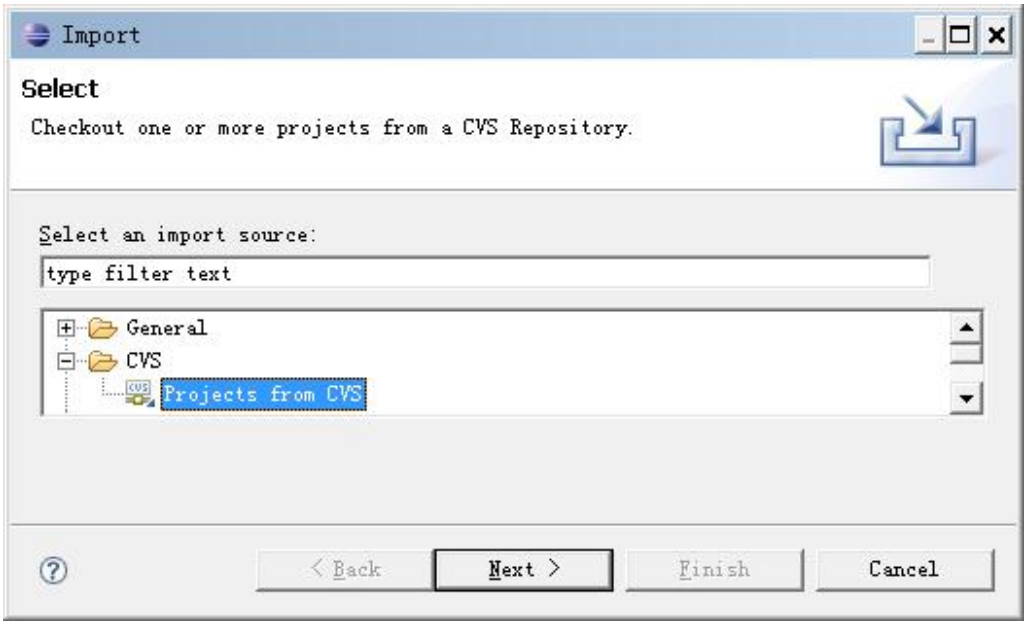


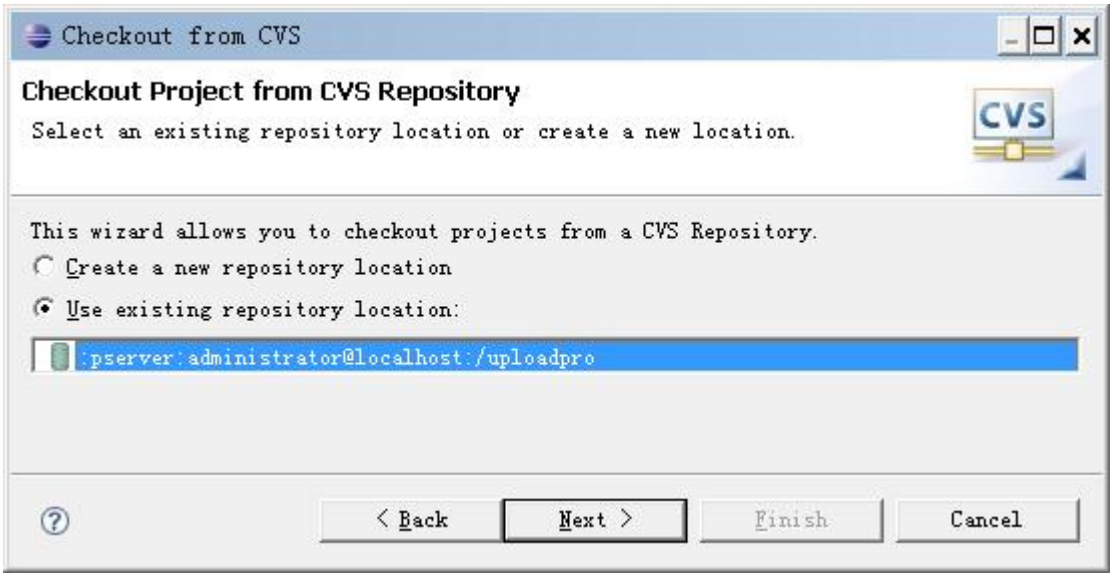



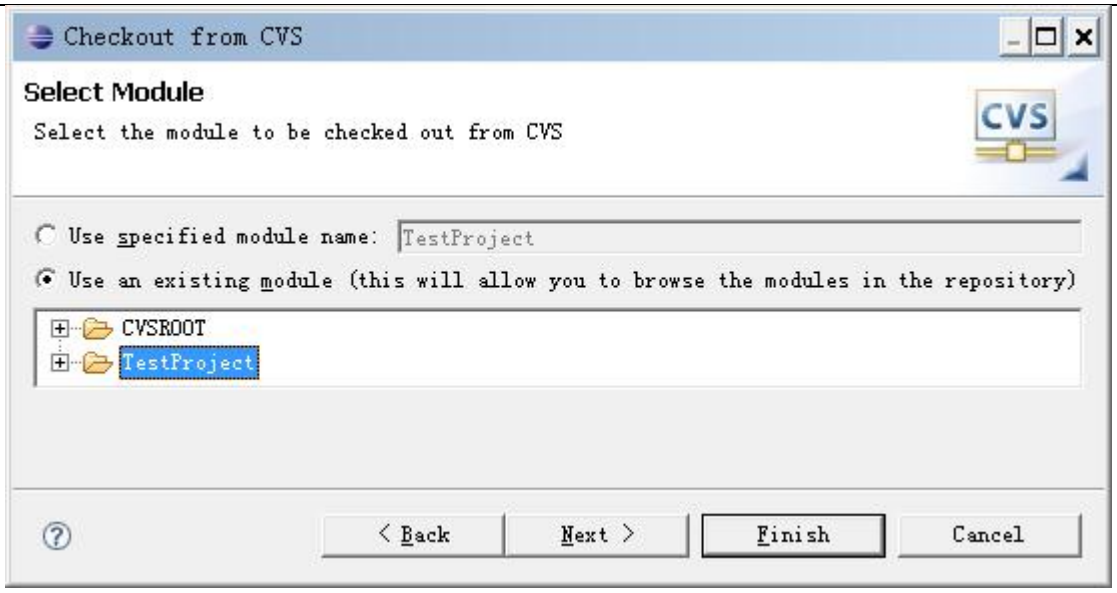



以后所有的项目可以直接从 CVS 服务器上下载。

文件 à 导入 à CVS




Checkout from CVS
Select Module
Select the module to be checked out from CVS
CVS
Use specified module name:
TestProject
Use an existing module (this will allow you to browse the modules in the repository)
CVSROOT
TestProject
< Back
Next >
Finish
Cancel

## 3.5、

| |
|---|

# 4、总结

# 5、预习任务

# 6、作业

| No. | 表达式 | 描述 |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| No. | 方法名称 | 类型 | 描述 |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |