Ant 文件目录

目录名	目的
项目目录	包括配置文件 build.xml 和本表中其它所有子目录
src	包括项目源代码文件
test	包括项目测试代码
lib	包括项目中需要的库文件
dist	部署文件，可能为 jar 或 war 文件等

软件配置管理，简称SCM（Software Configuration Management），用于跟踪和控制软件变更。
软件配置管理包括：版本控制（version control）、变更管理和过程支持等。
版本控制的对象是软件开发过程中涉及的所有计算机文件，包括最为重要的源代码，以及库文件，图形资源文件，计划文档，需求文档，设计文档，测试文档等可以存储为计算机文件的有关资源。
由服务器和客户端组成
服务器使用项目“存储库（repository）”存储受版本控制的所有文件以及文件的完整修订历史。
通常一个项目会在服务器上建立一个存储库，项目所需文件全部存储在该存储库中。
该存储库具有独立的用户访问控制，确保具有授权的开发人员才可以进行相关操作。
服务器在存储同一文件的变更时往往会采用“增量存储”的方式，即只保留文件相继版本之间的差异，这样可以更加有效的存储文件多个版本。
白盒测试指当测试者知道程序的内部数据结构和算法，并且能够获取其具体源码时进行的测试。白盒测试检查程序逻辑，确定测试用例，覆盖尽可能多的代码和逻辑组合。
黑盒测试认为软件是一个“黑匣子”，测试人员完全不知道其内部实现。测试工程师根据需求规格说明书确定测试用例，测试软件的功能，而不需要了解程序的内部结构。
调试过程：
重现错误。
定位错误。非常复杂。对错误提出尽可能多的假设，并通过各种调试工具和方法确认错误可能出现的区域。
改正错误。在改正错误之前需要理解问题的本质，并理解整个程序。Bug是系统非预期的行为，那么在寻找bug的时候必须正确理解系统的行为，这样才能改正错误。
软件存在bug
验证（verification）：验证指确定某个工作阶段是否正确完成的过程，在每个工作阶段结束时进行。我们是否正确地构建了软件？（软件是否符合软件规格说明书）
确认（validation）：确认是在产品交付用户之前进行的深入细致的评估，它的目的是确定整个产品是否满足用户期望。我们是否构建了正确的软件？（软件是否是用户需要的）
软件测试定义为：
“使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别。”

测试用例（Test Case）是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序是否正确工作。使用测试用例，是给程序一个输入，将程序的输出与预想的输出进行比较，如果相同则程序运行正确，否则，程序有错误。
集成（integration）指将单独的软件构件合并成一个整体的软件开发活动。
系统测试关注整个系统的行为，在完整的系统上进行，测试其是否符合系统需求规格说明书。
集成测试（integration testing），也叫组装测试或联合测试，是在完成单元测试后，将单独模块组合成为子系统或系统时进行的测试，在系统测试前进行。
单元测试是在开发过程中由程序员进行的一种测试，它主要测试程序模块的正确性
测试时间选择。
自动化。
测试用例相互独立
最简单的白盒测试是语句覆盖（statement coverage），即设计一系列的测试用例，使得程序中所有的语句都会得到执行。但这样的测试不能保证测试到所有的分支。
改进的方法是分支覆盖（branch coverage），即设计一系列的测试用例，保证所有的分支都得到测试。
最复杂的语句测试是路径覆盖（path coverage），即设计一系列的测试用例，保证覆盖程序中所有的语句或它们的所有可能组合，这是最强的一种要求
Junit测试用例的写法：
1新建类，继承junit.framework.TestCase类；
• 2.定义需要的测试方法，这些方法的名字以test开头，例如testAdd(),testPut()等，返回值为void；
• 3.如果需要组合测试用例，可以定义teat suite。
• ① assertTrue/False ([String message,] boolean condition);
• ②fail ([String message]);
• ③assertEquals([String message,]Object expected, Object actual);
• ④ assertNotNull/Null ([String message,] Objectobj);
• ⑤ assertSame/NotSame([String message,]Object expected, Object actual);
• ⑥ failNotSame/failNotEquals(String message, Object expected, Object actual)

```
importData.BookInfo;
importData.Catalog;
importjunit.framework.TestCase;
public class Catalog   Testextends  TestCase {
public void testAddBookInfo(){
BookInfo booka =new  BookInfo("123456", "HeadFirstJava","小明","清华出版社",2010,true);
Catalog testCatalog=newCatalog();
testCatalog.addBookInfo("123456",
"Head
FirstJava","小明","清华出版社",2010,true);
assertTrue(booka.getBookInfoRecord().equals(testCatalog.searchBookInfo("123456").get
BookInfoRecord()));
```

```
 }
}
```
在编写一个通用的模板时，有以下几步：
新建模板类，继承自TestCase类；
向模板中添加需要使用的实例变量；
重写setUp()方法，用以实例化变量。
重写tearDown()方法，用以释放在变量资源
TestSuite是JUnit提供的一个用于批量运行测试用例的对象，是test的一种有效的组合方式。

```
TestSuite suite = new TestSuite();
suite.addTest(new BookTest ("testBookInCatalog "));
suite.addTest(new BookTest ("testBookNotInCatalog "));
TestResult result = suite.run();
```

自动提取套件的方式：

```
public static Test suite()
{
return new TestSuite(BookTest.class);
}
```

```
importjunit.framework.Test;
importjunit.framework.TestSuite;
public class TestAll{
  public static Test suite(){
    TestSuite suite=new TestSuite();
    suite.addTestSuite(CatalogTest.class);
    return suite;
  }
  public static void main(String args[]) {
    junit.textui.TestRunner.run(suite());
  }
}
```