Findbugs使用说明i

目录
[隐藏]

插件下载

- Findbugs For Intellij Idea插件下载

Findbugs&CheckStyle For Eclipse 插件下载

Findbugs使用文档

- FindBug错误修改中文说明

不是最新资料，最全的信息参考以下的“bug描述清单”：

- FindBugs的详细bug描述清单（英文版）

Findbugs使用手册（英文版）

Findbugs常见问题解决办法

Findbugs错误分类说明(2.0.2)

★ Bad practice 坏的实践(85条)

一些不好的实践，下面列举几个：

HE： 类定义了equals()，却没有hashCode()；或类定义 了equals()，却使用Object.hashCode()；或类定义了hashCode()，却没有equals()；

或类定义了hashCode()，却使用Object.equals()；类继承了equals()，却使用Object.hashCode()。
SQL：Statement 的execute方法调用了非常量的 字符串；或Prepared Statement是由一个非常量的字符串产生。
DE： 方法终止或不处理异常，一般情况下，异常应该被 处理或报告，或被方法抛出。

★ Correctness 正确性问题（143条）
可能导致错误的代码，下面列举几个：
NP： 空指针被引用；在方法的异常路径里，空指针被引 用；方法没有检查参数是否null；null值产生并被引用；null值产生并在方法的异常
路径被引用；传给方法一个声明为@NonNull的null参数；方法的返回值声明为@NonNull实际是null。
Nm： 类定义了hashcode()方法，但实际上并未覆盖父类Object的hashCode()；类定义了tostring()方法，但实际上并未覆盖父类Object的
toString()；很明显的方法和构造器混淆；方法名容易混淆。
SQL：方法尝试访问一个Prepared Statement的0索引；方法尝试访问一个ResultSet的0索引。
UwF：所有的write都把属性置成null，这样所有的读取都是null， 这样这个属性是否有必要存在；或属性从没有被write。

★ Internationalization 国际化（2条）
DM：当对字符串使用upper或lowercase方法，如果是国际的字符串，可能会不恰当的转换。
DM：依赖默认的编码。

★ Malicious code vulnerability 代码漏洞（15条）
如果代码公开，可能受到恶意攻击的代码，下面列举几个：
FI： 一个类的finalize()应该是protected，而不是public的。
MS：属性是可变的数组；属性是可变的Hashtable；属性应该是package protected的。

★ Multithreaded correctness 多线程的正确性
多线程编程时，可能导致错误的代码，下面列举几个：

ESync：空的同步块，很难被正确使用。
MWN：错误使用notify()，可能导致IllegalMonitorStateException异常；或错误的使用wait()。
No： 使用notify()而不是notifyAll()，只是唤 醒一个线程而不是所有等待的线程。
SC： 构造器调用了Thread.start()，当该类被继承可能会导致错误。

★ Performance 性能问题（27条）
可能导致性能不佳的代码，下面列举几个：
DM：方法调用了低效的Boolean的构造器，而应该用Boolean.valueOf(...)；用类似
Integer.toString(1) 代替new Integer(1).toString()；方法调用了低效的float的 构造器，应该用静态的valueOf方法。
SIC：如果一个内部类想在更广泛的地方被引用，它应该声明为static。
SS： 如果一个实例属性不被读取，考虑声明为static。
UrF：如果一个属性从没有被read， 考虑从类中去掉。
UuF：如果一个属性从没有被使用，考虑从类中去掉。

★ Dodgy 潜在危险性代码（71条）
具有潜在危险的代码，可能运行期产生错误，下面列举几个：
CI： 类声明为final但声明了protected的属性。
DLS：对一个本地变量赋值，但却没有读取该本地变量；本地变量赋值成null，却没有读取该本地变量。
ICAST： 整型数字相乘结果转化为长整型数字，应该将整型 先转化为长整型数字再相乘。
INT：没必要的整型数字比较，如X <= Integer.MAX_VALUE。
NP： 对readline()的直接引用，而没有判断是否null； 对方法调用的直接引用，而方法可能返回null。
REC：直接捕获Exception，而实际上可 能是RuntimeException。
ST： 从实例方法里直接修改类变量，即static属性。

★ Security 关于代码安全性防护（11）

★ Experimental （3）

Findbugs eclipse插件使用说明

目的

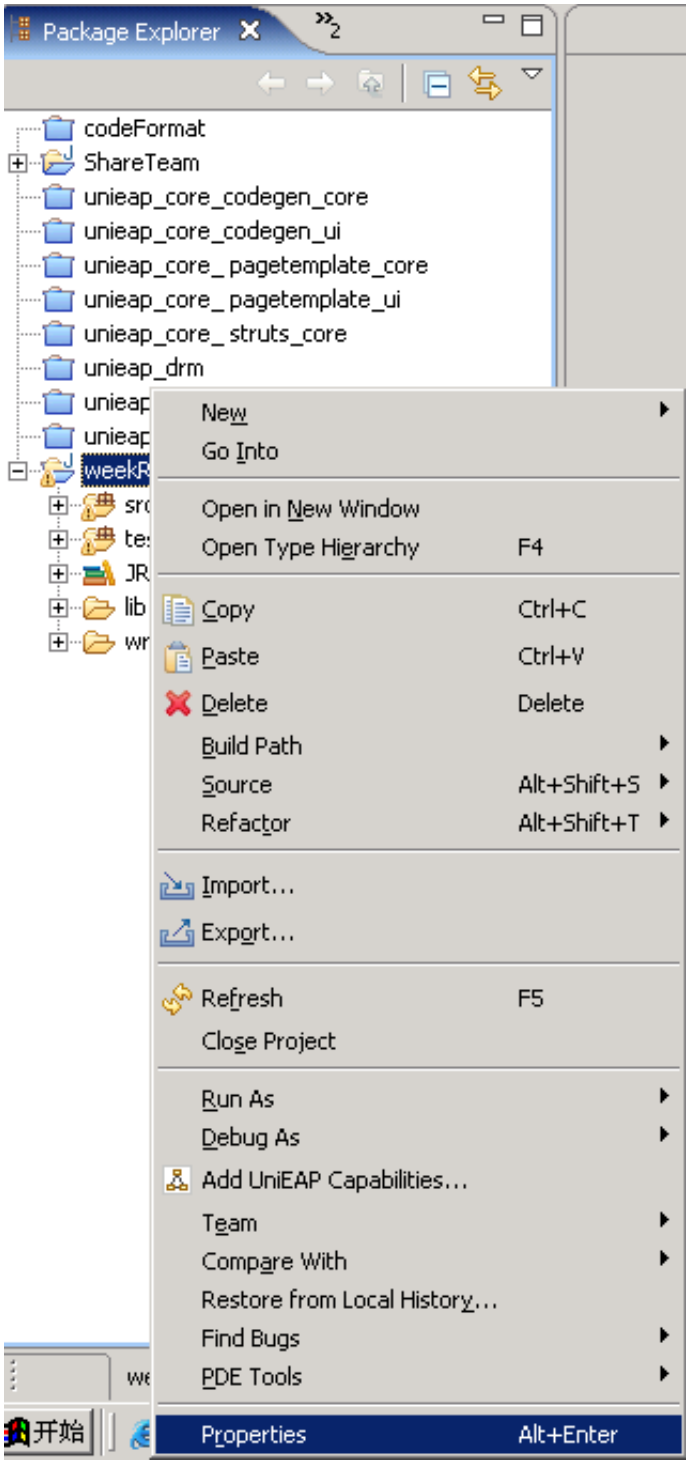
Findbugs是一个代码质量工具；我们用它来检查源代码中出现的伪问题，以期尽可能在项目的初始阶段将代码问题解决。本文主要介绍Findbugs的eclipse插件的应用。

概要

FindBugs 是一个静态分析工具，它检查类或者 JAR 文件，将字节码与一组缺陷模式进行对比以发现可能的问题。我们利用它在eclipse中的插件来对它所过滤的工程的源代码进行检查。希望在程序员编写代码的过程中，将代码中的缺陷指出来，让编码人员在开发中将它们纠正。达到尽可能在项目编码中将问题解决得目的。而不是在编码结束的时候才用该软件对代码检查，修改。

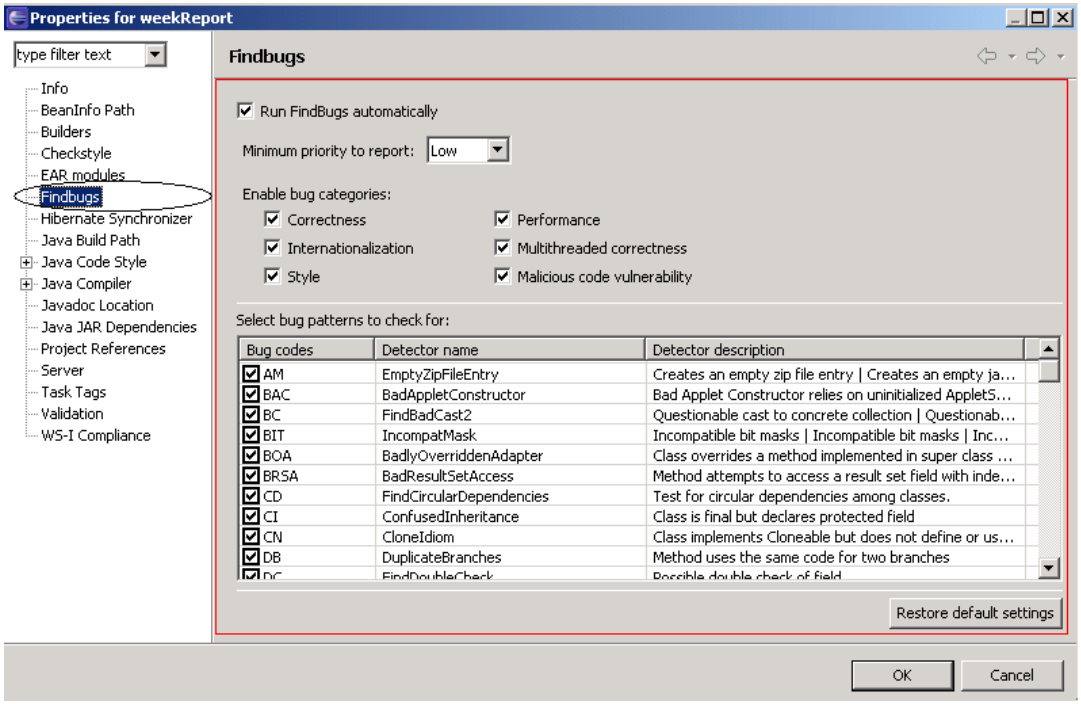
Findbugs使用说明

安装好Findbugs后我们启动eclipse。在eclipse中选择某个工程的属性。如下图所示：



Findbugs eclipse 1.png

点击属性弹出属性对话框，我们选择对话框左边的树上的“Findbugs”节点：



Findbugs eclipse 2.png

下面我们对Findbugs各项属性的配置进行一下说明：

★Run Findbugs automnaticaly：编译工程和文件的时候自动运行

★Minimum priority to report： 根据bug的优先权级别报告bug。

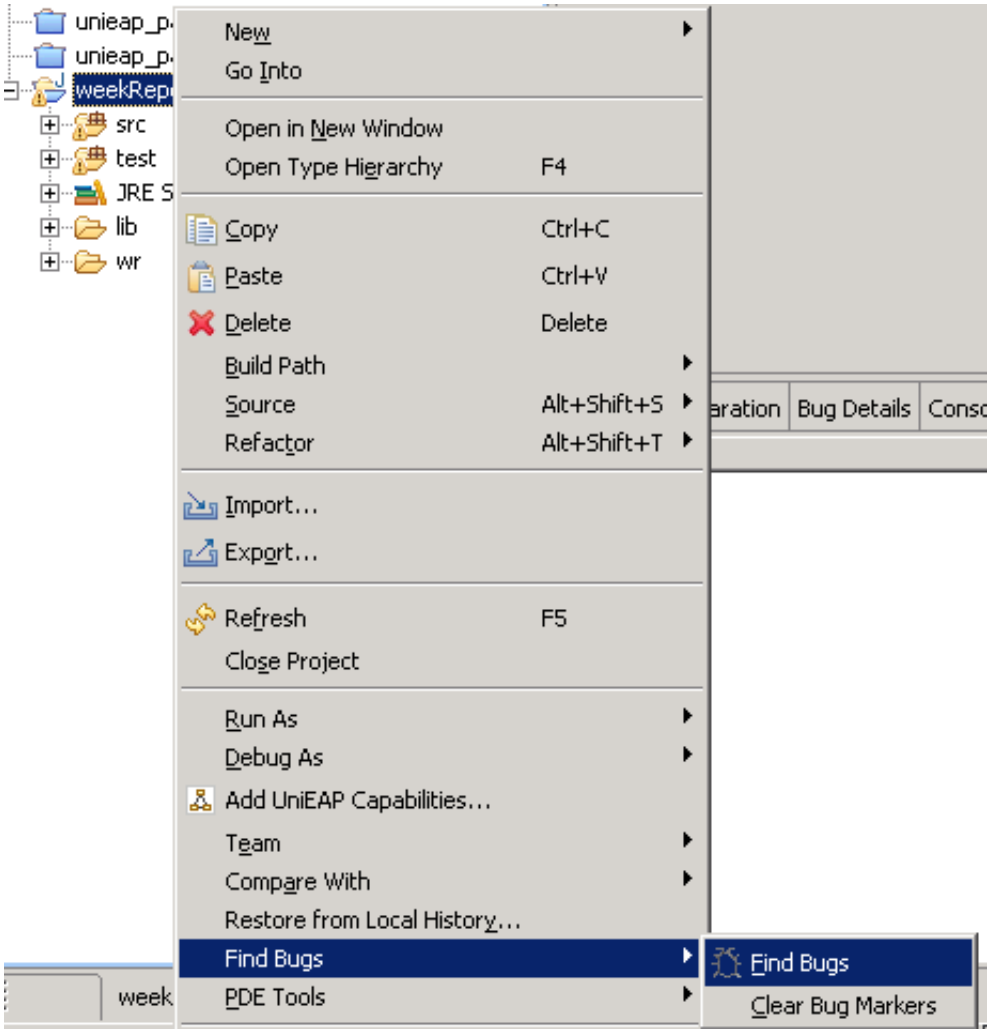
★ Enable bug categories: bug种类（[见以上文档]）。

★Select bug patterns to check for： bug的校验模式

Bug的校验模式的设置是确定哪一类问题我们应该作为bug报告给用户；

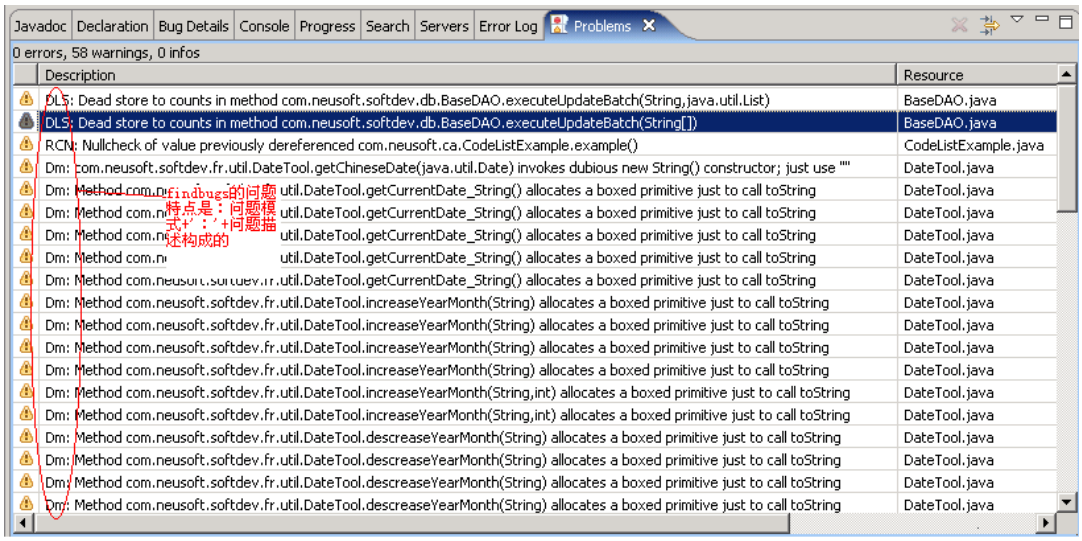
当我们根据部门规范选择要校验的模式后。点击”ok”按钮我们就可以对工程进行校验了。

如下图：我们在工程的菜单中选择findbugs的菜单项“Find bugs”。



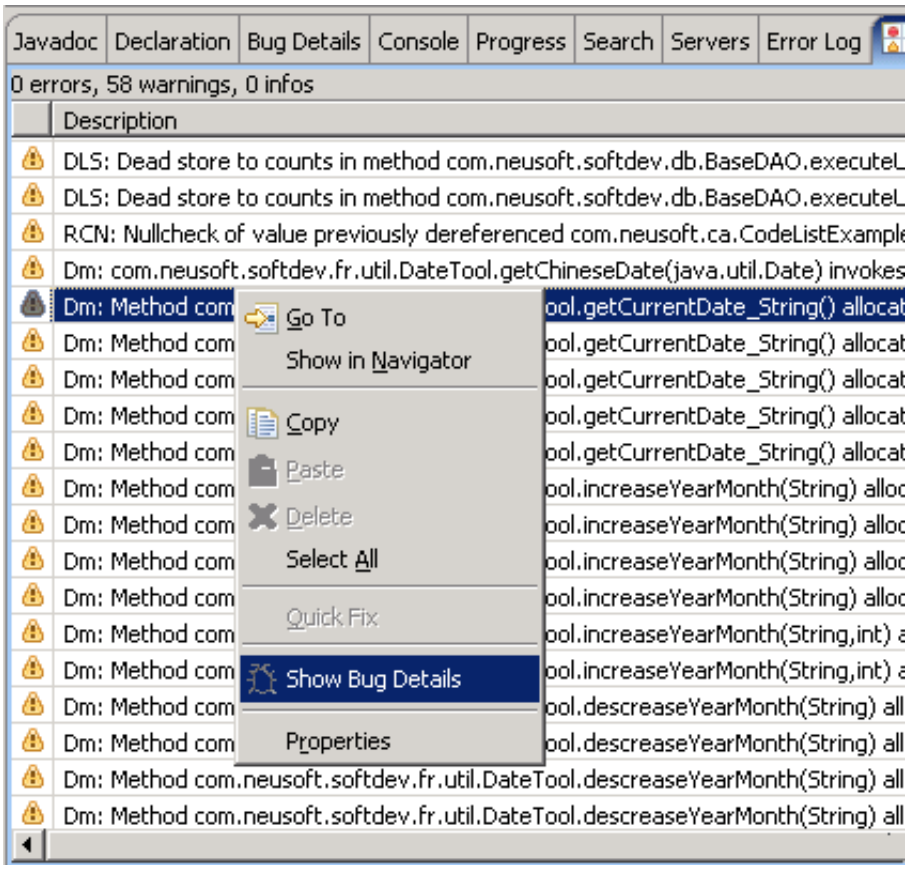
Findbugs eclipse 3.png

运行后我们可以在problems视图中看到工程的所有可能的“伪问题”

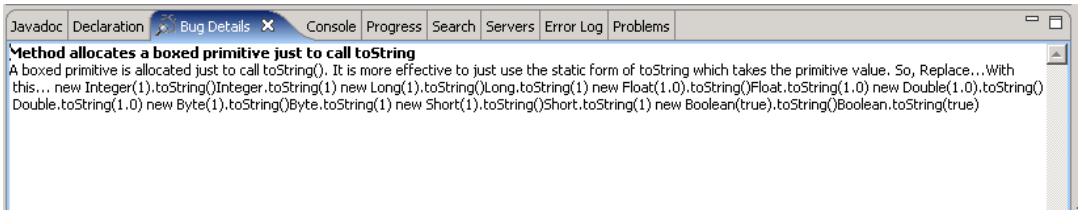


Findbugs eclipse 4.png

Findbugs的问题描述一般都是：问题类型+‘：’+问题描述构成的。如果我们想看到详细的问题描述，可以选择问题然后点击右键菜单，在“bug Details”视图中看到相应的说明。如下图所示：

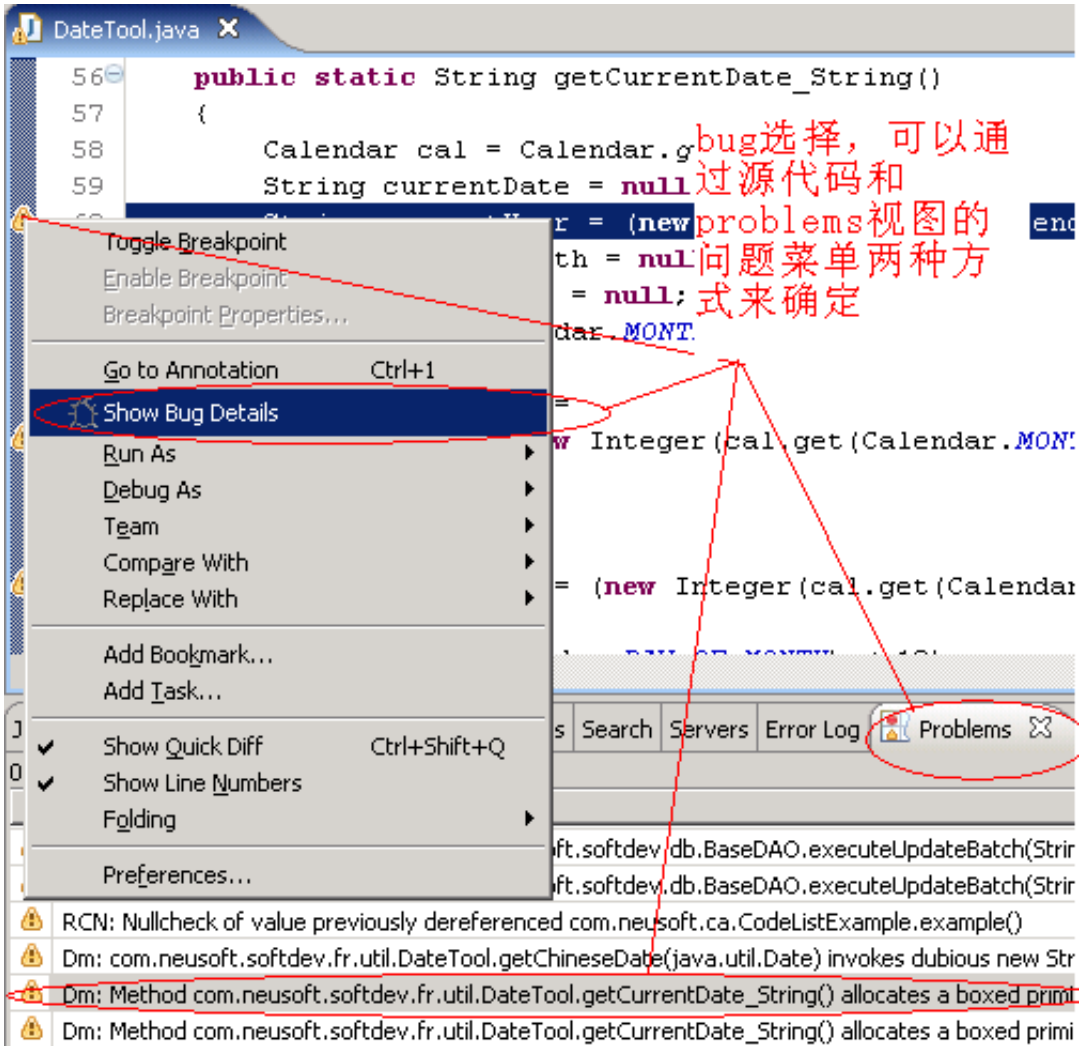


Findbugs eclipse 5.png



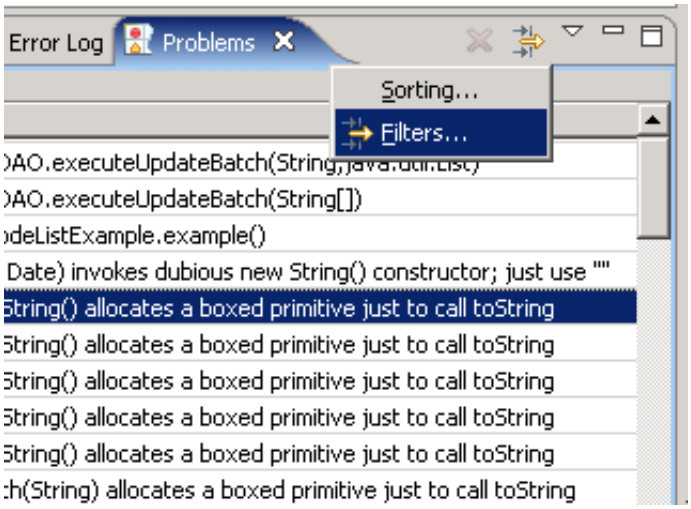
Findbugs eclipse 6.png

Bug选择的其他方式：

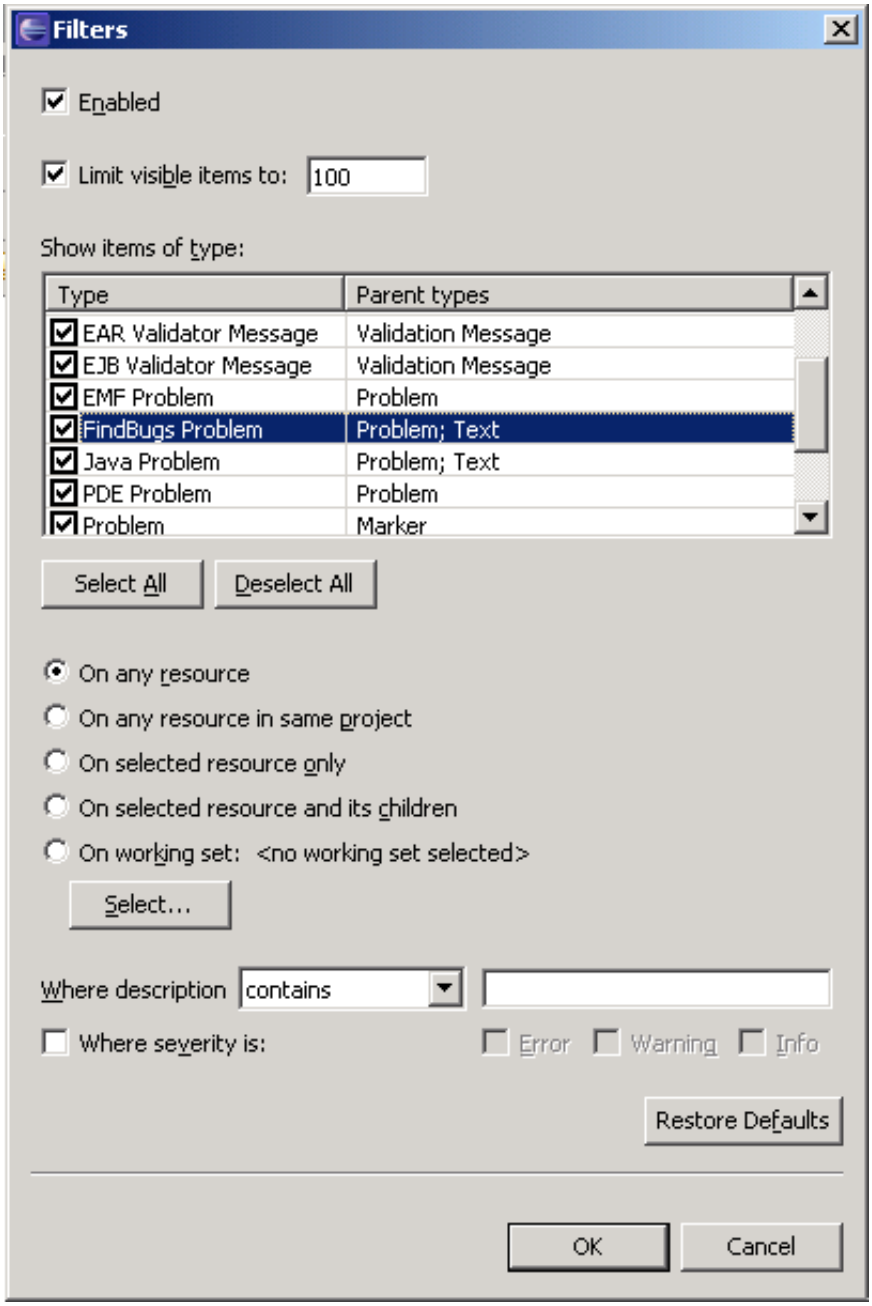


Findbugs eclipse 7.png

另外，如果我们想看到仅仅是findbugs检查出来的问题的话我们可以对问题列表进行过滤。如下图所示的方法可以实现问题过滤：



Findbugs eclipse 8.png



Findbugs eclipse 9.png