The background features three large, overlapping blue circles of varying sizes. Two thin blue lines intersect diagonally across the page, one from the top-left to the bottom-right, and another from the top-right to the bottom-left.

黑冰老师

QQ: 123512897

Eclipse 常用七大插件 介绍

黑冰

2016/10/31

Ecclipse 常用七大插件介绍.....	2
一. 七大插件概述.....	2
二. 七大插件详述	2
1. Findbugs.....	2
1) findbug 是什么	2
2) Findbugs 如何安装.....	3
a) Market 搜索安装.....	3
b) 插件 URL 安装.....	4
3) Findbugs 如何使用	6
2. CheckStyle.....	9
1) Checkstyle 是什么.....	9
2) checkstyle 安装.....	9
3) checkstyle 如何使用	9
3. Coverlipse.....	9
1) Coverlipse 是什么	9
2) Coverlipse 安装.....	9
3) Coverlipse 如何使用	10
4. CPD.....	11
1) CPD 是什么	11
2) CPD 安装	11
3) CPD 如何使用	11
5. Jdepend.....	13
1) Jdepend 是什么	13
2) Jdepend 安装	13
3) Jdepend 如何使用	13
4) 结果解析.....	13
6. Metrics	14
1) Metrics 是什么	14
2) Metrics 安装	14
3) 如何使用 Metrics	14
7.MAT.....	16
1) MAT 是什么	16
2) MAT 安装.....	16
3) MAT 使用	18
a) 直接观察 heap.....	19
b) 获取 dump 文件，分析文件.....	20

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

Eclipse 常用七大插件介绍

Eclipse 作为开发常用的工具,尤其 Android 开发测试几乎都基于 Eclipse 环境。在 Android 开发和自动化测试中,有一些方便的插件可以使用,本文档介绍七个常用的插件。

插件的安装主要有三分方式:

- ✓ 在 Eclipse 市场中搜索直接安装(最方便)。
- ✓ 加载插件对应的 URL 安装(大众化)。
- ✓ 下载插件包,放到 Eclipse 的 Plugin 目录下(很少用)。

黑冰老师

QQ: 123512897

一. 七大插件概述

工具	作用	RUL	备注
Findbugs	探查代码缺陷	http://findbugs.cs.umd.edu/eclipse	
CheckStyle	编码标准分析	http://eclipse-cs.sourceforge.net/update	与tint工具类似
Coverlipse	代码覆盖率	http://coverlipse.sf.net/update	Cobertura的Eclipse插件
CPD	复制/粘贴检验(重复率)	http://pmd.sourceforge.net/eclipse	
JDepend	包依赖项分析(架构质量)	http://andrei.gmxhome.de/eclipse/	
Metrics	复杂度监控	http://metrics.sourceforge.net/update	
MAT	动态分析内存	见文档中安装说明	

二. 七大插件详述

1. Findbugs

1) findbug 是什么

findbugs 是一个开源的 eclipse 代码静态检查工具;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。它可以简单高效全面地帮助我们发现程序代码中存在的 bug, bad smell, 以及潜在隐患。针对各种问题,它并且提供了简单的修改意见供我们重构时进行参考;通过使用它,可以一定程度上降低我们 code review 的工作量,并且会提高 review 效率。通过 findbugs 找到 bug,再由我们自己重构代码,可以培养我们的编码意识及水平,形成好的习惯提高开发编码能力。

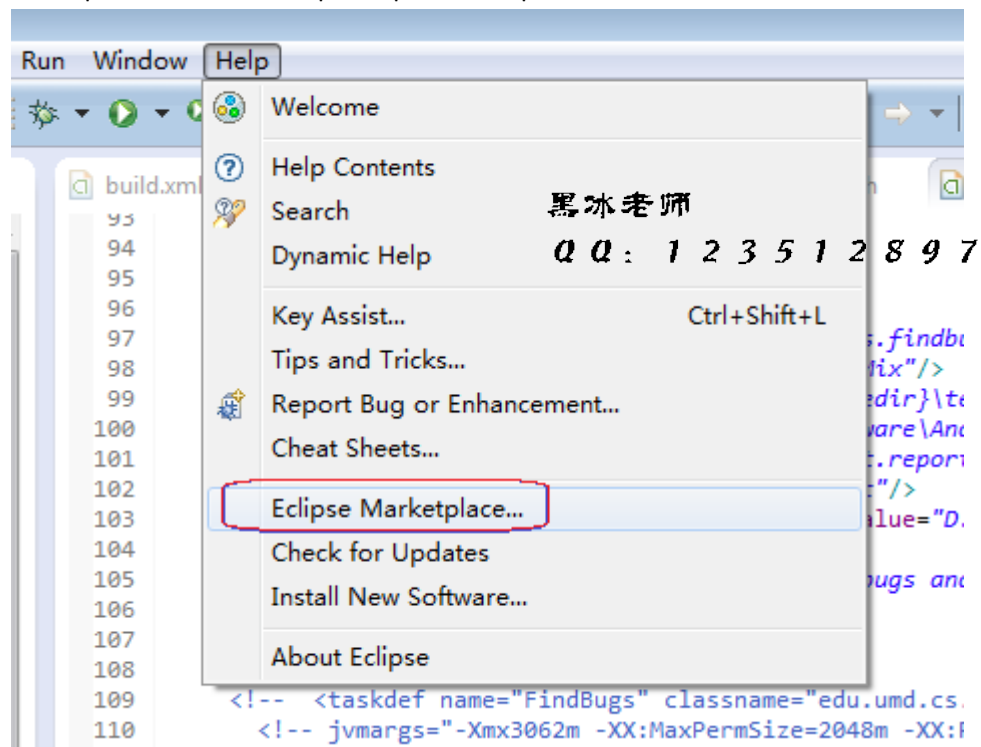
Findbugs 可以发现的 bug 种类,见官网详细列表

<http://findbugs.sourceforge.net/bugDescriptions.html>

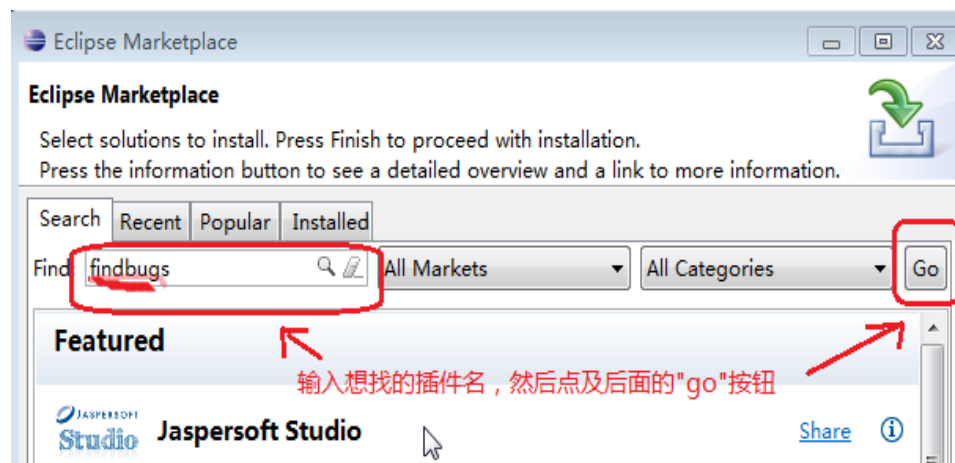
2) Findbugs 如何安装

a) Market 搜索安装

在 Eclipse 工具栏上点“help->Eclipse Marketplace”，如下图所示。



在弹出的 market 页面中，输入需要的插件名，然后点“搜索”按钮，如下图所示。

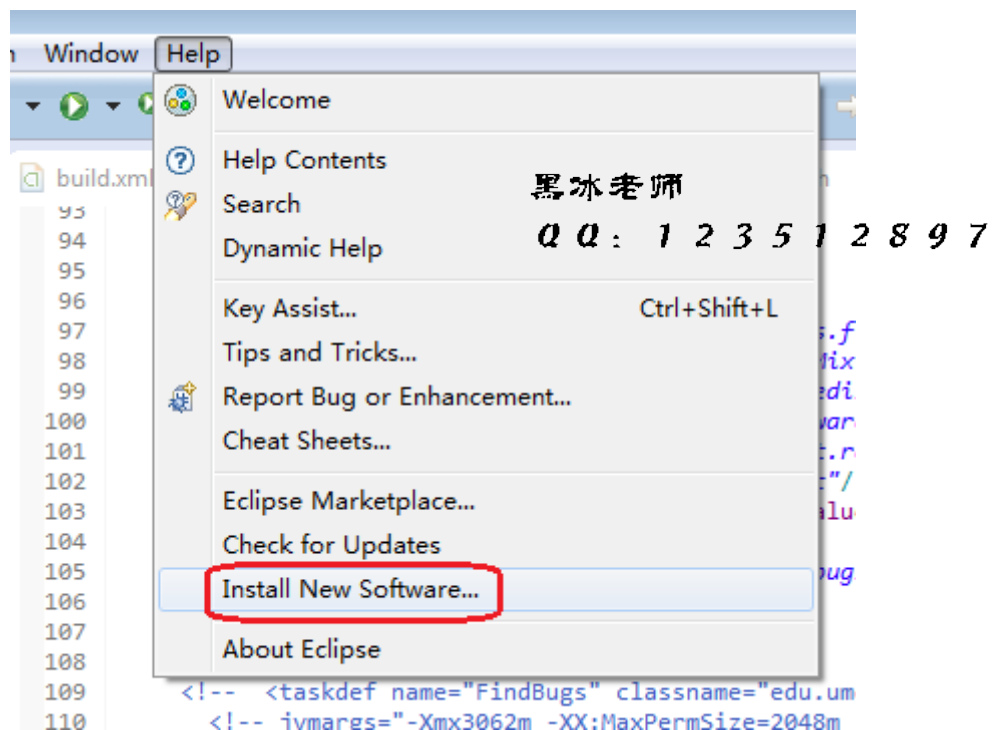


有可能会搜出多个相关的插件，比如谷歌原生的，第三方扩展的等等，选择我们想要的，点击其后的“install”按钮，开始安装，有需要接受相关协议的直接选接受，提示重启的选重启即可。如下图所示例。

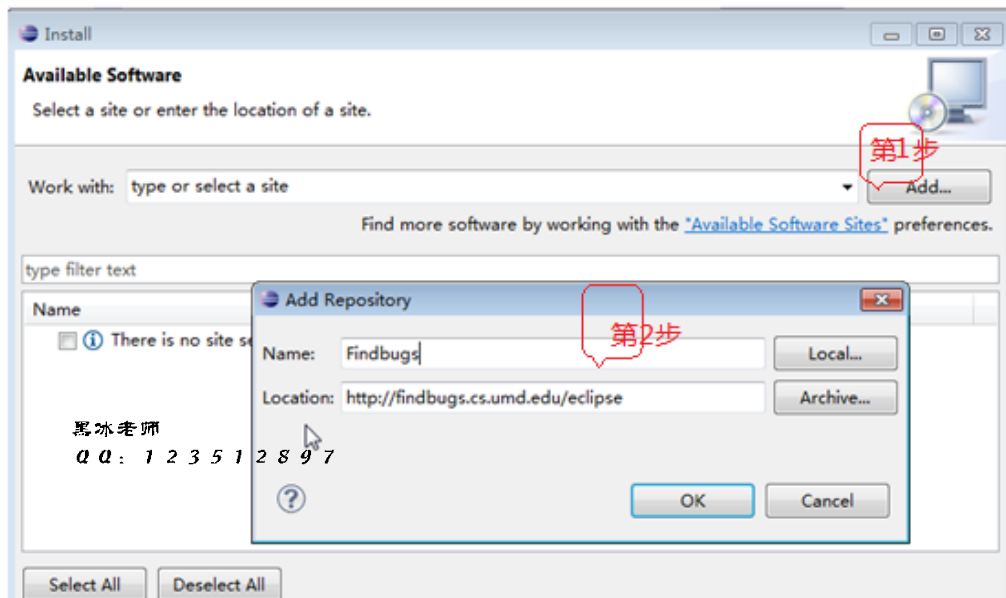


b) 插件 URL 安装

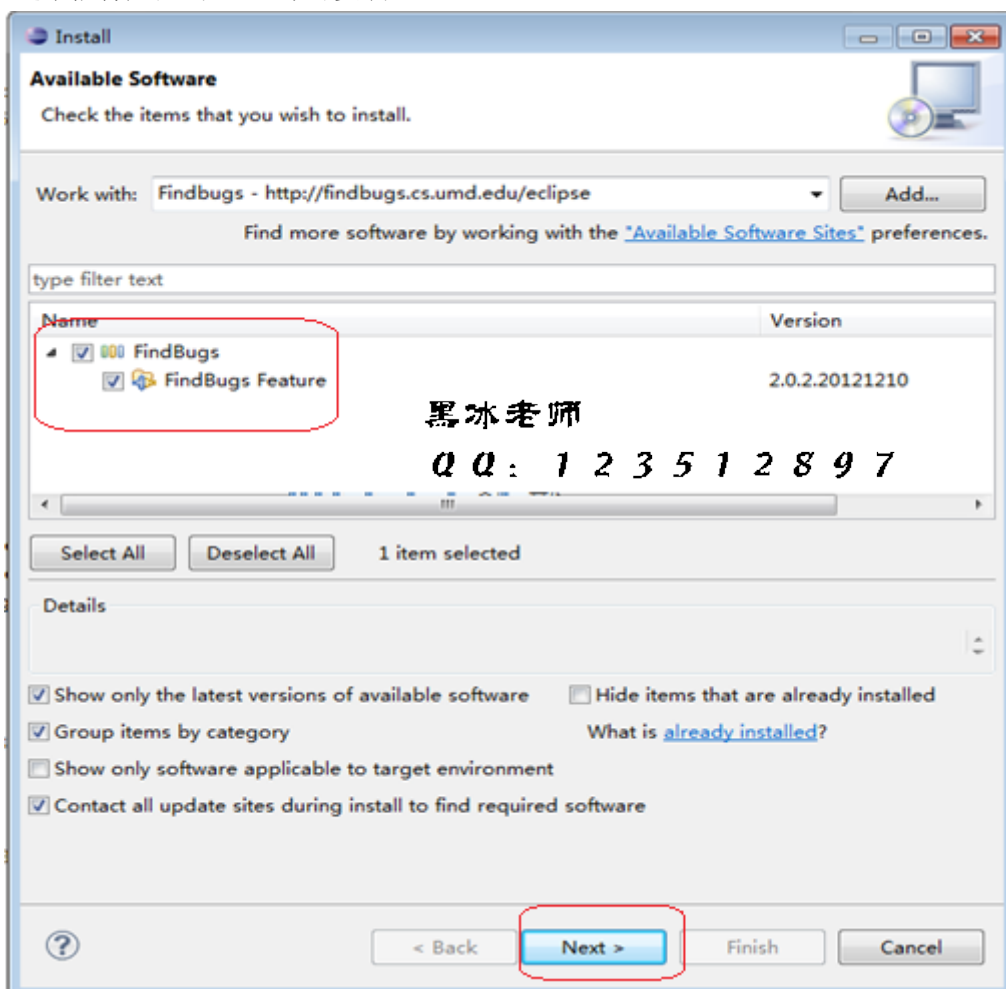
点击 Eclipse 工具栏上的 Help->Install New Software...



在弹出框中点“Add”按钮，在新弹出框中，有两项需要填写，Name 项自由填个名字即可，Location 项需要填写该插件对应的 URL，然后点“OK”按钮，如下图所示。

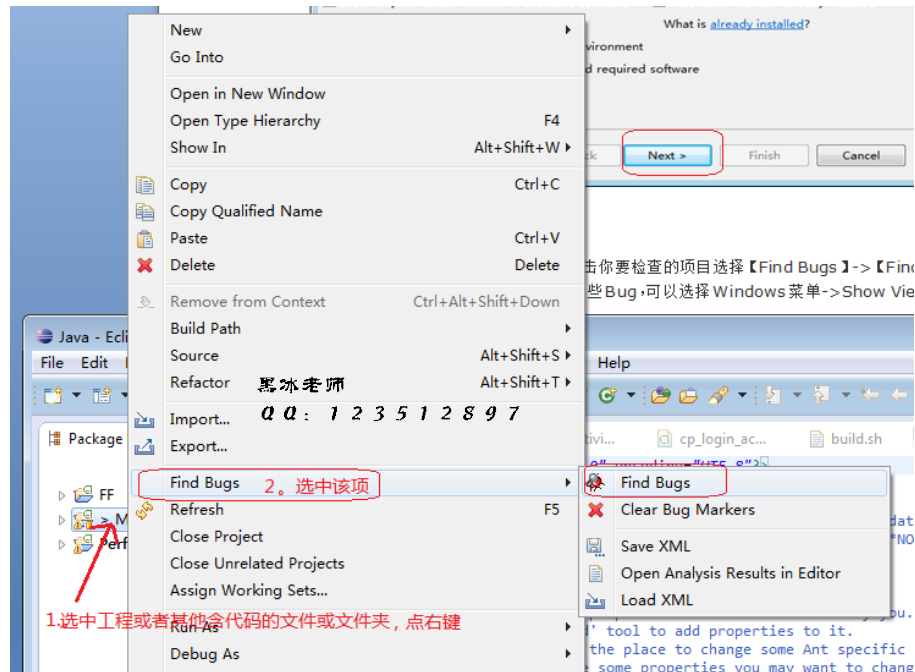


选中插件后点击 Next 即可安装。

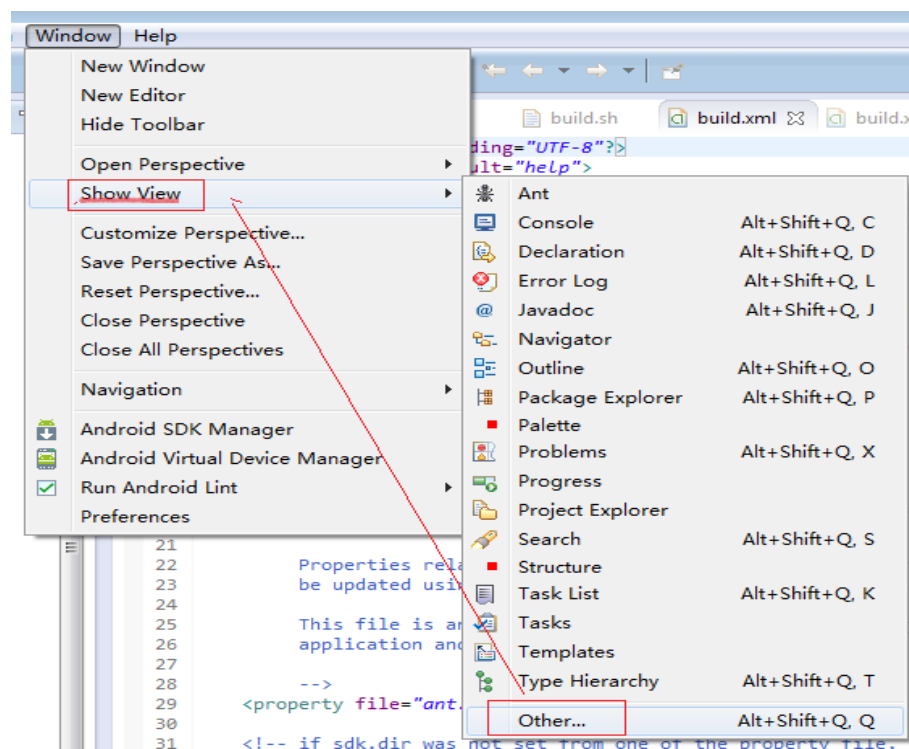


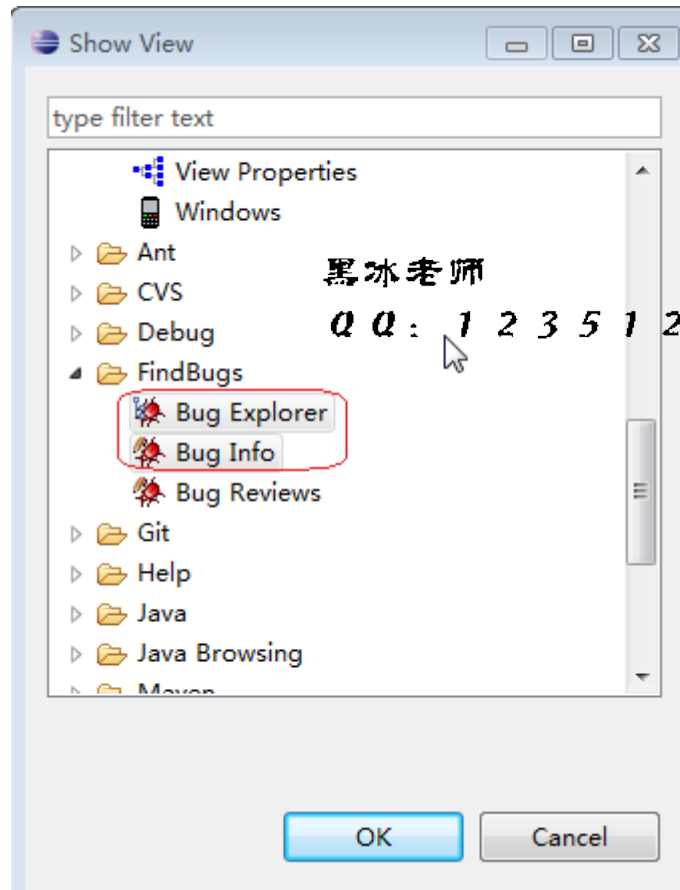
3) Findbugs 如何使用

安装了 Findbugs 插件后。右键点击你要检查的项目选择【Find Bugs】->【Find Bugs】进行检查。本文以彩票 Android 客户端代码为例，项目名是 Mix。

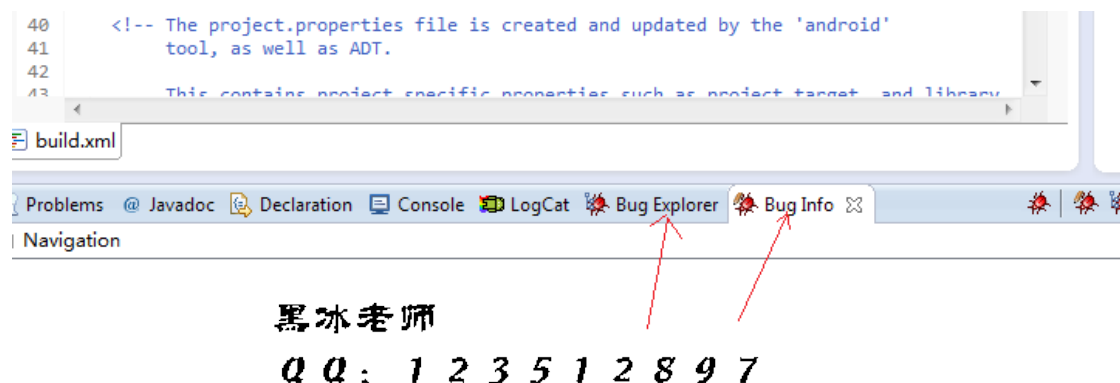


要查看 Findbugs 检查出了哪些 Bug，可以选择 Windows 菜单->Show View->other->Bug Explorer 及 Bug info，会在 Eclipsey 页面上打开 Bug Explorer 面板。

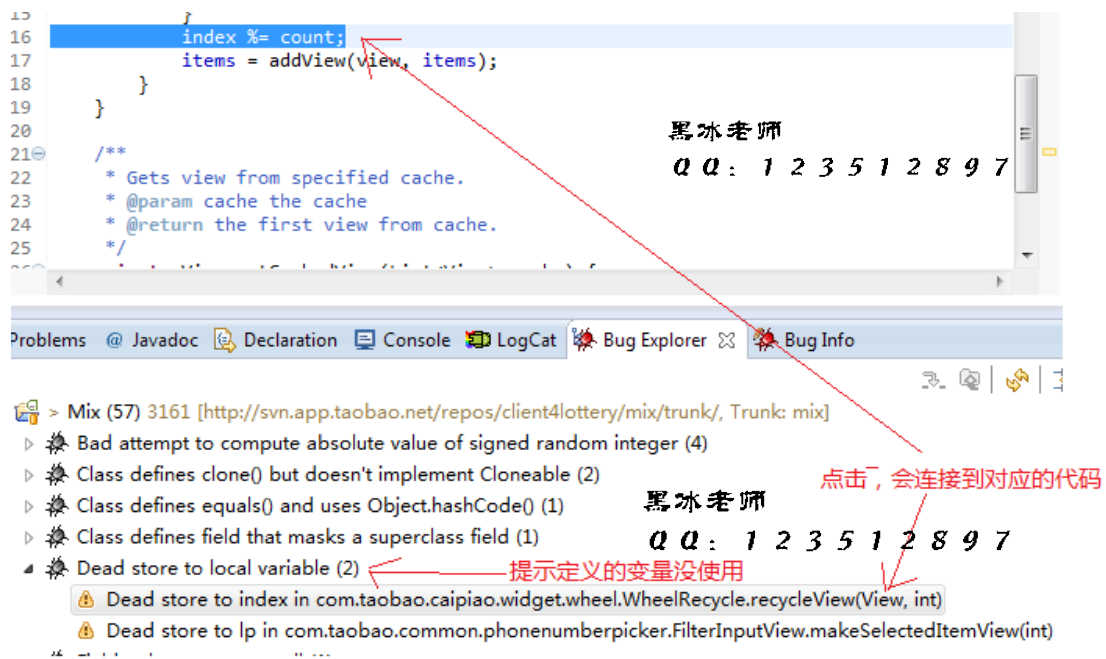




点击“OK”后，打开了 Bug Explore 和 Bug Info 窗口。



如果想要查看某个 Bug 详细的信息，则可以选择 Windows 菜单->Open Perspective，然后选择 FindBugs 就可以打开 FindBugs 的 Properties 面板，在这个面板里面可以看到最详尽的 Bugs 信息.以我们彩票 Android 客户端代码为例，在 Bug Explorer 窗口中展示结果，点击某个类别下的某条提示，会直接定位到对应的代码上，如下图所示。



点及 Bug Info, 在 Bug Info 窗口中显示详细的 bug 信息, 如下图。



黑冰老师
QQ: 1 2 3 5 1 2 8 9 7

2. CheckStyle

1) Checkstyle 是什么

Checkstyle 是检查代码风格的统一性，把所有代码和已经定义好的标准对比，不符合标准的代码会以 problem 抛出。CheckStyle 的检查规则包含在 XML 格式的配置文件里。CheckStyle 默认的 sun 规范检查文件是：sun_checks.xml

2) checkstyle 安装

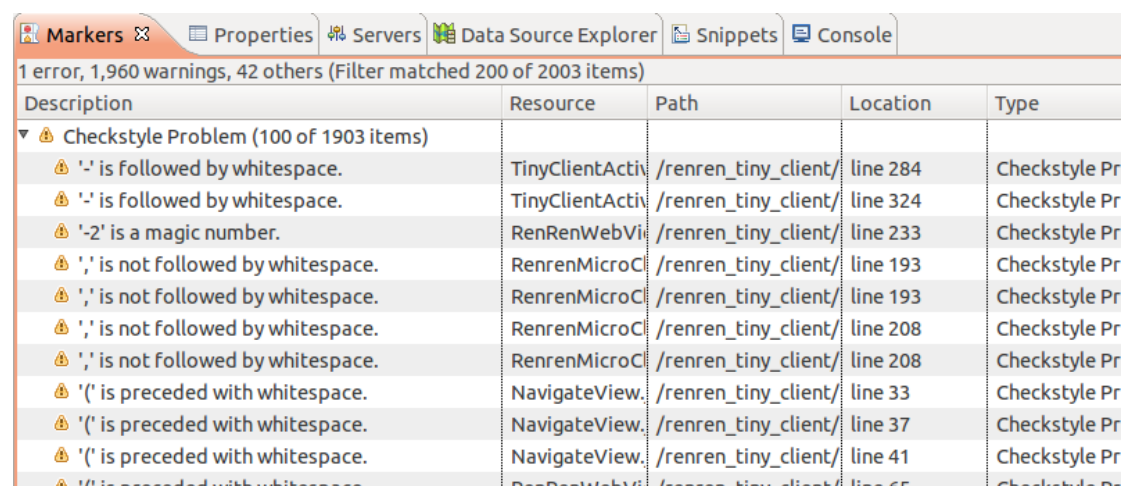
同 Findbugs 安装类似

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

3) checkstyle 如何使用

选目标代码（代码文件夹或者文件都可以），右键 Checkstyle->Check Code with Check style。



The screenshot shows the Eclipse IDE's Markers view. At the top, it says '1 error, 1,960 warnings, 42 others (Filter matched 200 of 2003 items)'. Below this is a table with columns: Description, Resource, Path, Location, and Type. The table lists several Checkstyle problems, such as ' is followed by whitespace.' and ' is not followed by whitespace.', with their respective file paths and line numbers.

Description	Resource	Path	Location	Type
Checkstyle Problem (100 of 1903 items)				
' ' is followed by whitespace.	TinyClientActiv	/renren_tiny_client/	line 284	Checkstyle Pr
' ' is followed by whitespace.	TinyClientActiv	/renren_tiny_client/	line 324	Checkstyle Pr
'-2' is a magic number.	RenRenWebVi	/renren_tiny_client/	line 233	Checkstyle Pr
';' is not followed by whitespace.	RenrenMicroC	/renren_tiny_client/	line 193	Checkstyle Pr
';' is not followed by whitespace.	RenrenMicroC	/renren_tiny_client/	line 193	Checkstyle Pr
';' is not followed by whitespace.	RenrenMicroC	/renren_tiny_client/	line 208	Checkstyle Pr
';' is not followed by whitespace.	RenrenMicroC	/renren_tiny_client/	line 208	Checkstyle Pr
'(' is preceded with whitespace.	NavigateView.	/renren_tiny_client/	line 33	Checkstyle Pr
'(' is preceded with whitespace.	NavigateView.	/renren_tiny_client/	line 37	Checkstyle Pr
'(' is preceded with whitespace.	NavigateView.	/renren_tiny_client/	line 41	Checkstyle Pr
'(' is preceded with whitespace.	RenRenWebVi	/renren_tiny_client/	line 65	Checkstyle Pr

3. Coverlipse

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

1) Coverlipse 是什么

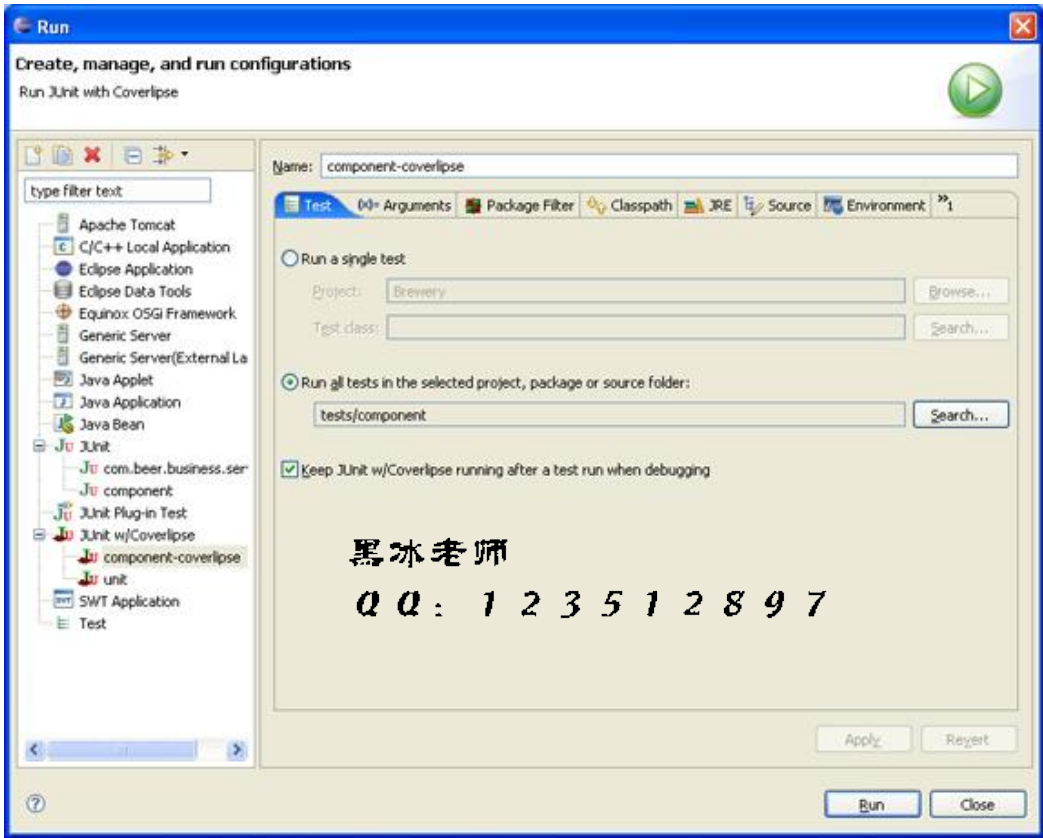
Coverlipse 是一个用于 Cobertura 的 Eclipse 插件，Cobertura 是一个代码覆盖率工具，可以用它来评估具有相应测试的源代码的比率。Cobertura 也提供一个 Ant 任务和 Maven 插件，但用 Cobertura，可以在编写代码时 评估代码覆盖率。

2) Coverlipse 安装

与 Findbugs 安装类似（最近几天 Coverlipse 官网无法连接，可稍后再装，或者直接百度个包，放到 Eclipse 的 plugin 目录下）。

3) Coverlipse 如何使用

右键单击测试工程并选择 **JUnit w/Coverlipse** 节点中的 **New**。如果右键点击没出现 Junit w/Coverlipse 则选 run as -> run configuration 便会出现 **JUnit w/Coverlipse**，右键点击 **JUnit w/Coverlipse** 选择 **New**，需要确定 JUnit 测试的位置。



一旦单击了 **Run**，Eclipse 会运行 Coverlipse 并在源代码中嵌入标记，该标记显示了具有相关 JUnit 测试的代码部分。

行覆盖率如下：

Content Description of CoverageMarkerView				
Message	Line	covered	uses	uncover
✓ This line was fully covered	4			
✓ This line was fully covered	8			
✓ This line was fully covered	12			
! This line was not covered	16			
✓ This line was fully covered	20			
✓ This line was fully covered	21			
! This line was not covered	24			
✓ This line was fully covered	28			

黑冰老师
QQ: 123512897

类覆盖率（会把测试代码的覆盖率也统计，这是 bug，我们只看功能代码的覆盖率即可）如下：



4. CPD

1) CPD 是什么

CPD 可以检查代码的重复率（复制/粘贴检查）。

2) CPD 安装

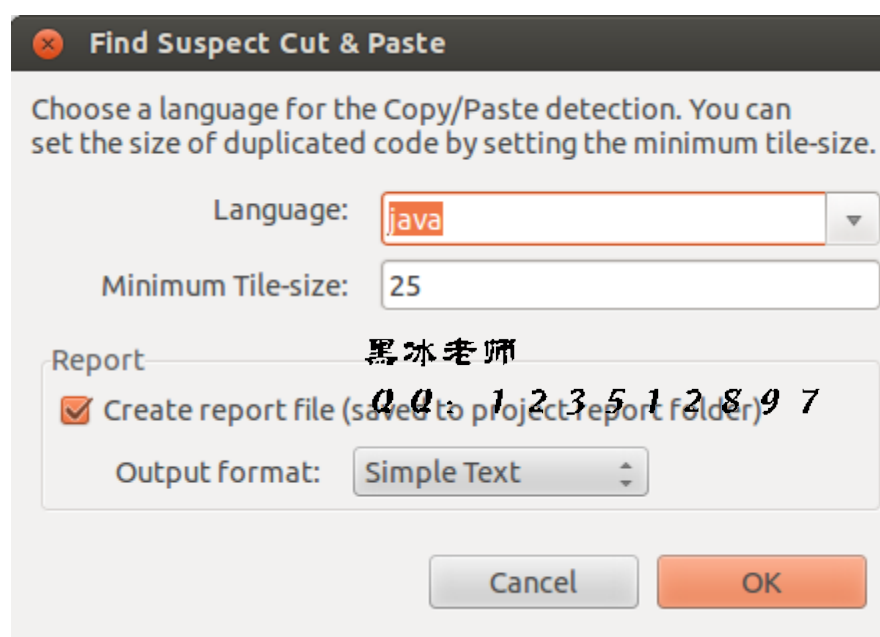
与 Findbugs 类似。

黑冰老师

Q Q : 1 2 3 5 1 2 8 9 7

3) CPD 如何使用

选择一个 Eclipse 项目单击右键，选择 PMD -> Find Suspect Cut and Paste，设置认定重复条件等。



点击 OK 后，生成结果。

Violations Overview		CPD View	Violations Outline
	Message	Class	
✖	Found suspect cut & paste (2 matches,17 lines)		
📄	lines 38-54 in file LoginResponse.java	src.com.renren.mobile.rm	
📄	lines 32-48 in file GetLoginInfoResponse.java	src.com.renren.mobile.rm	
✖	Found suspect cut & paste (2 matches,12 lines)		
✖	Found suspect cut & paste (2 matches,12 lines)		
✖	Found suspect cut & paste (2 matches,13 lines)		
✖	Found suspect cut & paste (2 matches,15 lines)		
✖	Found suspect cut & paste (2 matches,9 lines)		
✖	Found suspect cut & paste (2 matches,15 lines)		

会在工程根路径下生成 report 文件夹。

rmsdk_core_impl

src

gen [Generated Java Files]

Android 2.1

Android Dependencies

Referenced Libraries

assets

bin

libs

reports

cpd-report.txt

res

AndroidManifest.xml

ant.properties

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

里面有 cpd_report.txt 报告文件，可点击察看。

```
1 Found a 17 line (104 tokens) duplication in the following files:
2 Starting at line 38 of /home/xb/workspace1/rmsdk_core_impl/src/com/renren
3 Starting at line 32 of /home/xb/workspace1/rmsdk_core_impl/src/com/renren
4
5     public GetLoginInfoResponse(
6         @JsonProperty("uid") long uid,
7         @JsonProperty("session key") String sessionKey,
8         @JsonProperty("ticket") String ticket,
9         @JsonProperty("secret_key") String secretKey,
10        @JsonProperty("user_name") String userName,
11        @JsonProperty("head_url") String headUrl,
12        @JsonProperty("now") long now,
13        @JsonProperty("login_count") long loginCount,
14        @JsonProperty("fill_stage") long fillStage) {
15        super();
16        this.uid = uid;
17        this.sessionKey = sessionKey;
18        this.ticket = ticket;
19        this.secretKey = secretKey;
20        this.userName = userName;
```

5. Jdepend

1) Jdepend 是什么

JDepend 是为包依赖项提供面向对象的度量值，以此指明代码库的弹性。换句话说，JDepend 可有效测量一个架构的健壮性（反之，脆弱性）。

除了 Eclipse 插件，JDepend 还提供一个 Ant 任务、Maven 插件和一个 Java 应用程序，用以获取这些度量值。对于相同的信息，它们有着不同的传递机制；但 Eclipse 插件的特别之处和相应优点是：它能以更接近源代码（即，编码时）的方式传递这条信息。

2) Jdepend 安装

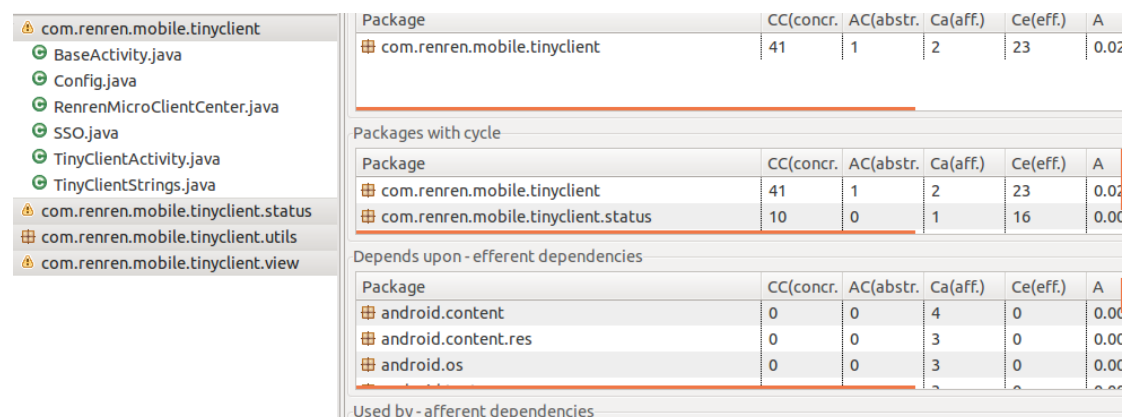
类似安装。

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

3) Jdepend 如何使用

通过右键单击源文件夹并选择 **Run JDepend Analysis**。一定要选择一个含源代码的源文件夹；否则看不到此菜单项。



Package	CC(concr.)	AC(abstr.)	Ca(aff.)	Ce(eff.)	A
com.renren.mobile.tinyclient	41	1	2	23	0.02

Package	CC(concr.)	AC(abstr.)	Ca(aff.)	Ce(eff.)	A
com.renren.mobile.tinyclient	41	1	2	23	0.02
com.renren.mobile.tinyclient.status	10	0	1	16	0.00

Package	CC(concr.)	AC(abstr.)	Ca(aff.)	Ce(eff.)	A
android.content	0	0	4	0	0.00
android.content.res	0	0	3	0	0.00
android.os	0	0	3	0	0.00

Package	CC(concr.)	AC(abstr.)	Ca(aff.)	Ce(eff.)	A
android.content	0	0	4	0	0.00
android.content.res	0	0	3	0	0.00
android.os	0	0	3	0	0.00

4) 结果解析

a) Number of Classes (Cc)

被分析 package 的具体和抽象类（和接口）的数量，用于衡量 package 的可扩展性。如果一个类中实现了其他类，如实现了监听类，则监听类的数目也记录在此。

b) Afferent Couplings (Ca)

依赖于被分析 package 的其他 package 的数量，用于衡量 package 的职责。即有多少包调用了它。

c) Efferent Couplings (Ce)

被分析 package 的类所依赖的其他 package 的数量，用于衡量 package 的独立性。即它调用了多少其他包。

黑冰老师

d) Abstractness (A)

QQ: 1 2 3 5 1 2 8 9 7

被分析 package 中的抽象类和接口与所在 package 所有类数量的比例，取值范围为 0—1。

e) Instability (I)

$I = Ce / (Ce + Ca)$ ，用于衡量 package 的不稳定性，取值范围为 0—1。I=0 表示最稳定，I=1 表示最不稳定。

即如果这个类不调用任何其他包，则它是最稳定的。

f) Distance (D)

被分析 package 和理想曲线 $A+I=1$ 的垂直距离，用于衡量 package 在稳定性和抽象性之间的平衡。理想的 package 要么完全是抽象类和稳定 ($x=0, y=1$)，要么完全是具体类和不稳定 ($x=1, y=0$)。

取值范围为 0—1，D=0 表示完全符合理想标准，D=1 表示 package 最大程度地偏离了理想标准。即你的包要么全是接口，不调用任何其他包（完全是抽象类和稳定），要么是具体类，不被任何其他包调用。

6. Metrics

1) Metrics 是什么

使用该插件可以进行许多有用的代码度量，包括圈复杂度度量，它用于测量方法中唯一路径的数目。

2) Metrics 安装

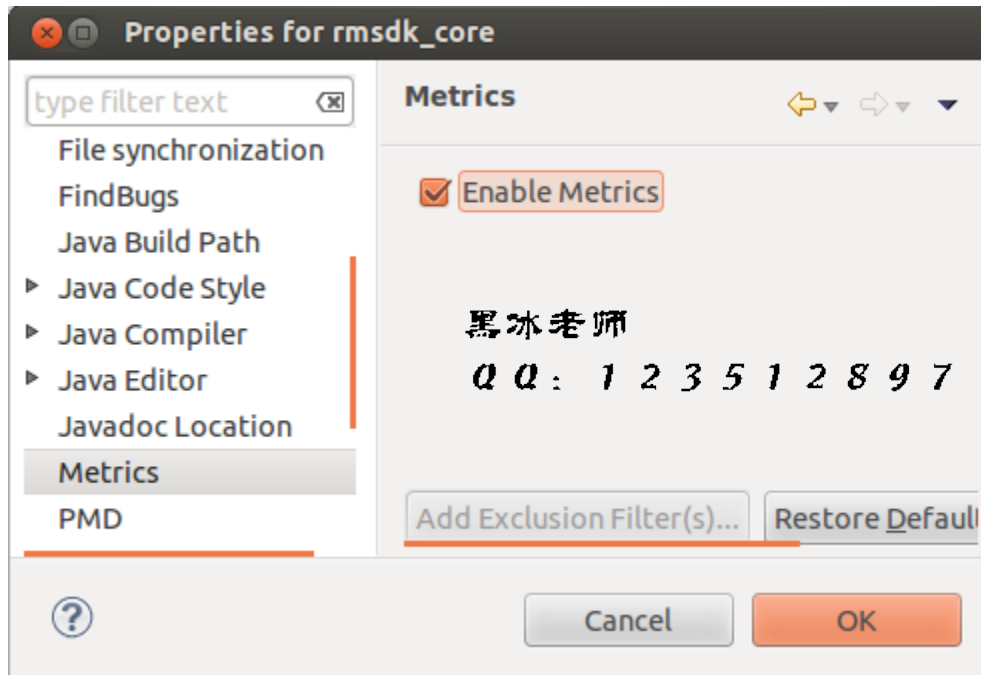
参考 FindBugs 安装。

黑冰老师

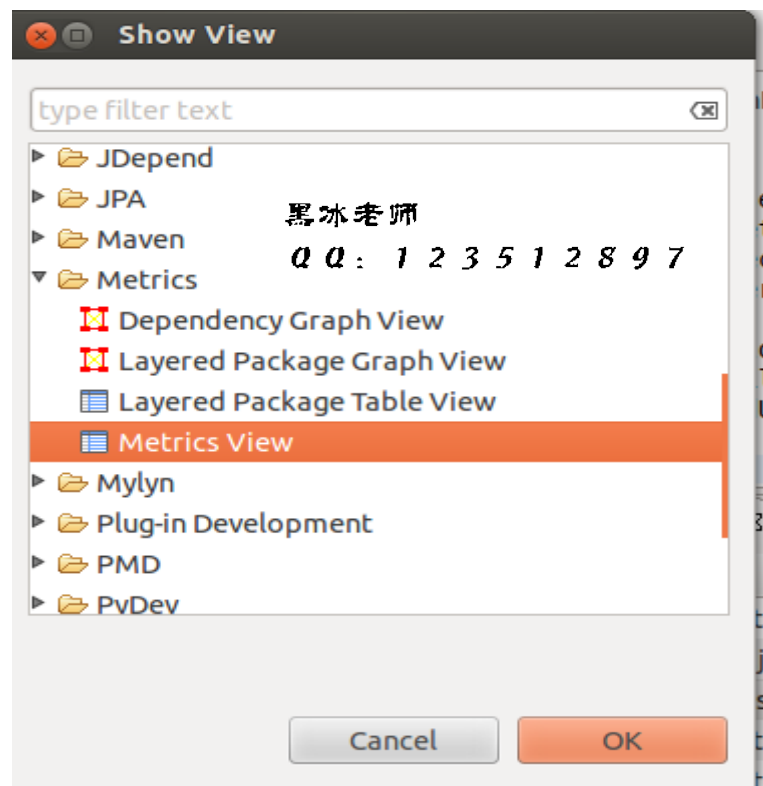
QQ: 1 2 3 5 1 2 8 9 7

3) 如何使用 Metrics

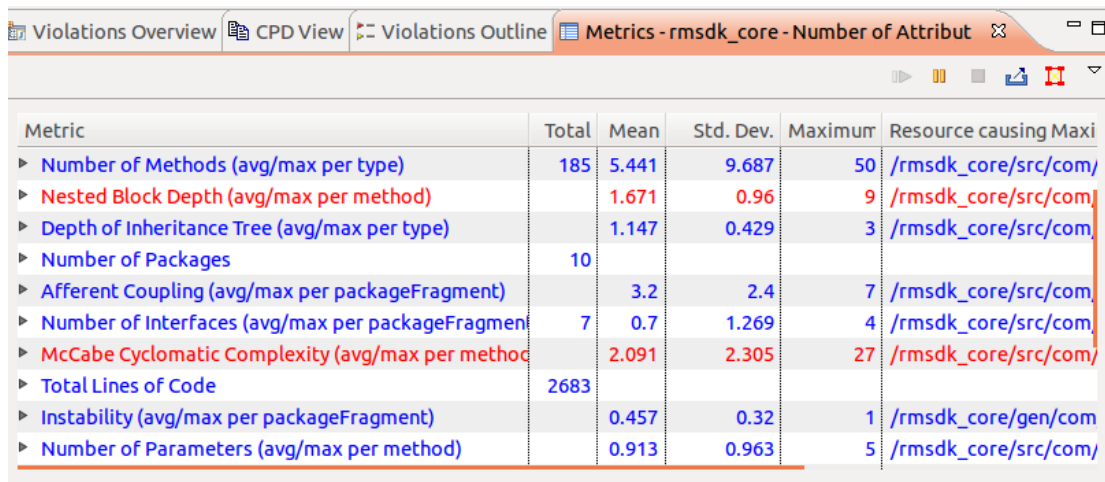
a) 右键单击您的项目并选择 Properties 菜单。在结果窗口中，选择 Metrics->Enable Metrics plugin 复选框并单击 “OK”。



- b) 从 Eclipse 中选择 Window 菜单打开 Metrics 视图，然后选择 Show View | Other...。
- c) 选择 Metrics | Metrics View 您需要使用 Java 透视图并重新构建项目，从而显示这些度量值。



在此例中，我正在查看一个单独方法的圈复杂度。可以双击 **Metrics** 列表中的方法，该插件会在 **Eclipse** 编辑器中为此方法打开源代码。这就让修正变得超级简单（如果需要）！



Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum
▶ Number of Methods (avg/max per type)	185	5.441	9.687	50	/rmsdk_core/src/com/
▶ Nested Block Depth (avg/max per method)		1.671	0.96	9	/rmsdk_core/src/com/
▶ Depth of Inheritance Tree (avg/max per type)		1.147	0.429	3	/rmsdk_core/src/com/
▶ Number of Packages	10				
▶ Afferent Coupling (avg/max per packageFragment)		3.2	2.4	7	/rmsdk_core/src/com/
▶ Number of Interfaces (avg/max per packageFragment)	7	0.7	1.269	4	/rmsdk_core/src/com/
▶ McCabe Cyclomatic Complexity (avg/max per method)		2.091	2.305	27	/rmsdk_core/src/com/
▶ Total Lines of Code	2683				
▶ Instability (avg/max per packageFragment)		0.457	0.32	1	/rmsdk_core/gen/com
▶ Number of Parameters (avg/max per method)		0.913	0.963	5	/rmsdk_core/src/com/

黑冰老师

7.MAT

QQ: 1 2 3 5 1 2 8 9 7

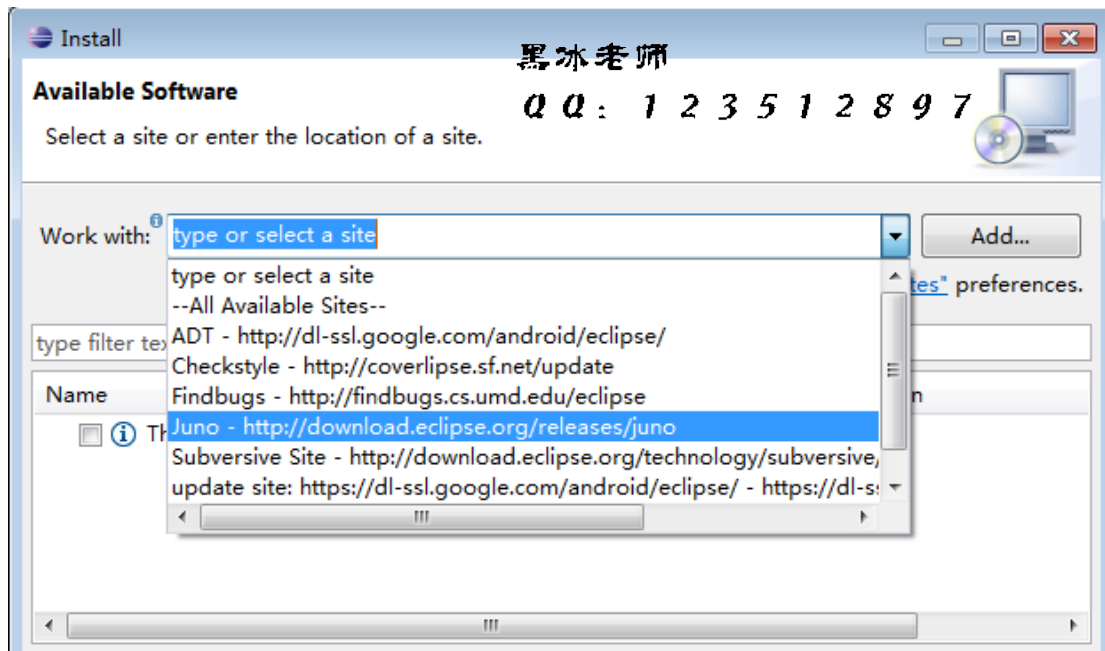
1) MAT 是什么

如果使用 **DDMS** 确实发现了我们的程序中存在内存泄漏，那又如何定位到具体出现问题的代码片段，最终找到问题所在呢？如果从头到尾的分析代码逻辑，那肯定会把人逼疯，特别是在维护别人写的代码的时候。这里介绍一个极好的内存分析工具 -- **Memory Analyzer Tool(MAT)**。MAT 与 **FindBugs** 结合起来使用效果更好。

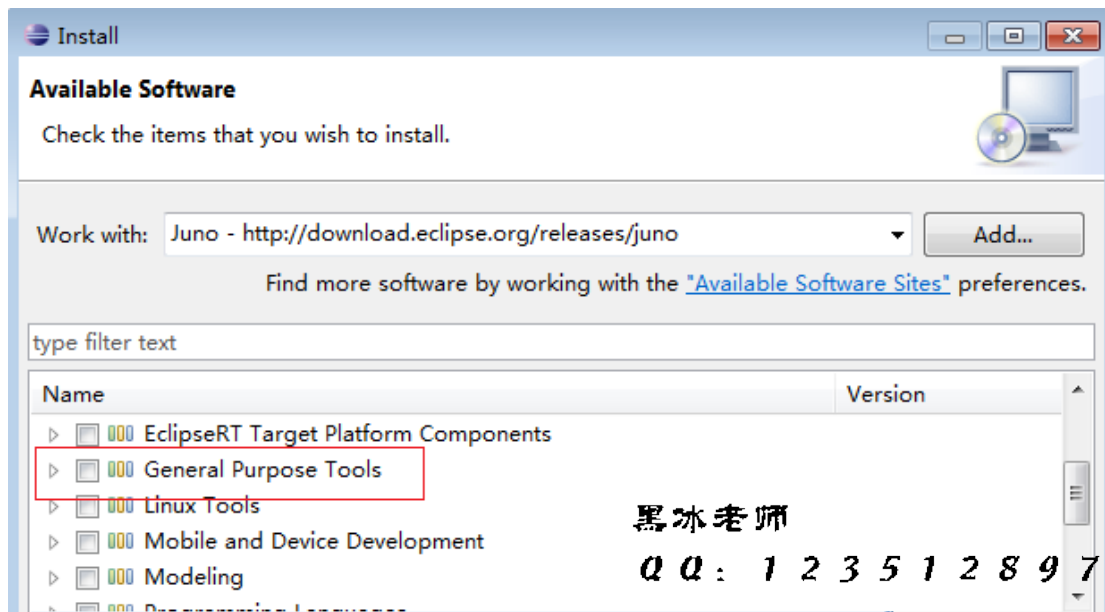
MAT 是一个 **Eclipse** 插件，同时也有单独的 **RCP** 客户端。官方下载地址、MAT 介绍和详细的使用教程请参见：www.eclipse.org/mat，在此不进行说明了。另外在 MAT 安装后的帮助文档里也有完备的使用教程。在此仅举例说明其使用方法。我自己使用的是 MAT 的 **eclipse** 插件，使用插件要比 **RCP** 稍微方便一些。

2) MAT 安装

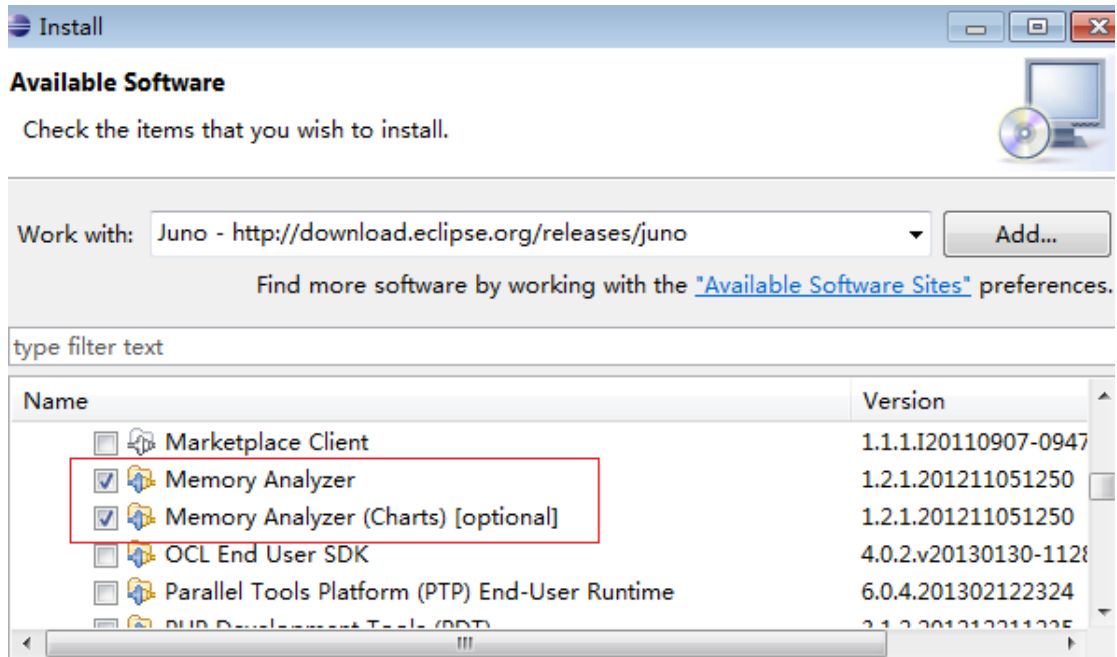
Eclipse 工具栏点击 **help->Install New Software**，在弹出框中，点击”**Work with**”对应的下拉框，从中选择 **Eclipse** 对应的 URL，比如我装的 Eclipse 是 **juno** 版本，所以我选择 **Juno** - <http://download.eclipse.org/releases/juno> 项，如下图所示。



点击“OK”确认后，出现可下载插件列表。



展开“General purpose Tools”。选中“Memory Analyzer”及“Memory Analyzer(Charts) [optional]”两项，点“next”开始安装，安装需要很长时间，请耐心等待。



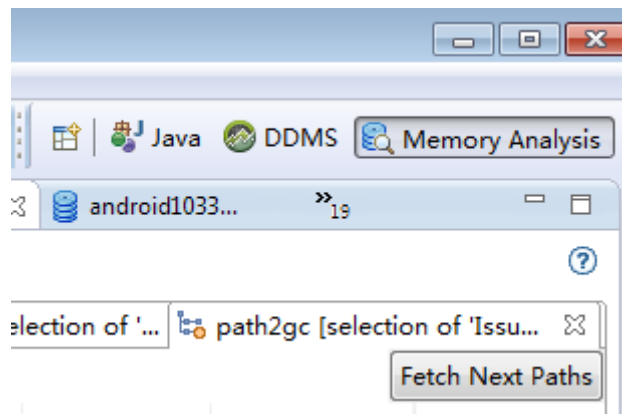
黑冰老师

3) MAT 使用

QQ: 1 2 3 5 1 2 8 9 7

MAT 使用有两种方式，一种是直接观察分析 heap 变换。另一种是获取 dump hprof 文件分析该文件。

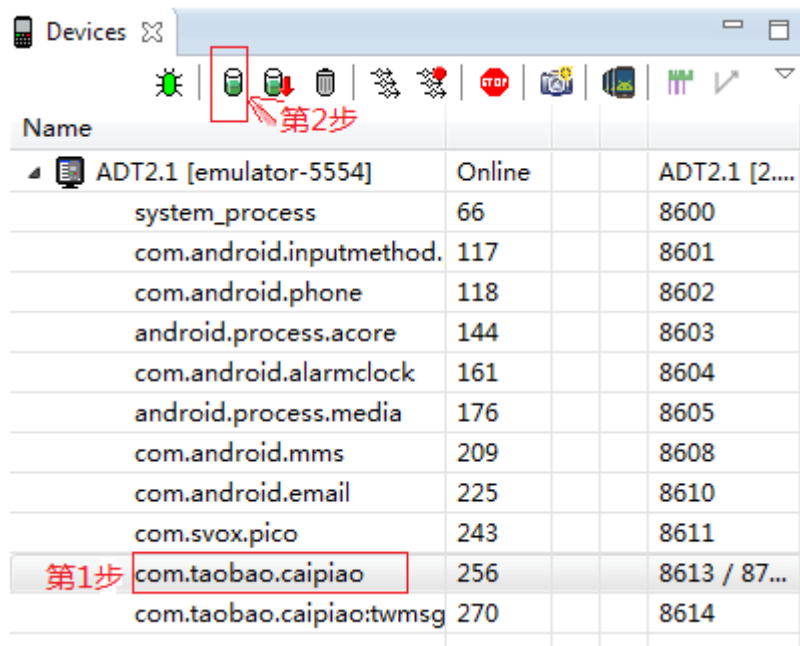
以彩票 Android 客户端代码为例，分别示例直接观察 heap 变化和获取 dump 文件分析。启动模拟器或者真机上的彩票应用程序(真机确保 USB 连接并且是 USB 调试模式)，Eclipse 切换到 DDMS 界面。



在 DDMS 启动下，Eclipse 左侧窗口显示 devices，如下图所示。

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7



Name			
ADT2.1 [emulator-5554]	Online		ADT2.1 [2....
system_process	66		8600
com.android.inputmethod.	117		8601
com.android.phone	118		8602
android.process.acore	144		8603
com.android.alarmclock	161		8604
android.process.media	176		8605
com.android.mms	209		8608
com.android.email	225		8610
com.svox.pico	243		8611
com.taobao.caipiao	256		8613 / 87...
com.taobao.caipiao:twmsg	270		8614

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

a) 直接观察 heap

选中某个进程，点“Update heap”也就是上图左上角的绿色图标。

右侧页面中，点开 Heap 窗口，如下图所示。点击一次页面上的 Cause GC 按钮，该按钮是清空 heap,相当于测试值初始化。然后开始操作真机或者模拟器，比如我在操作竞猜足球的选场次->投注->删除所选场次的循环操作。在操作中需要重点关注两个数据的变换，一个是 heap 大小的变化，另一个是使用总内存 data object 的大小变化，如下图所示。



ID	Heap Size	Allocated	Free	% Used	# Objects	
1	6.195 MB	4.594 MB	1.601 MB	74.15%	77,132	Cause GC

Type	Count	Total Size	Smallest	Largest	Median	Average
free	4,005	1.567 MB	16 B	244.141 KB	56 B	411 B
data object	51,665	2.297 MB	16 B	736 B	32 B	411 B
class object	3,347	576.016 KB	176 B	184 B	176 B	176 B
1-byte array (byte[], boolean[])	1,000	382.617 KB	24 B	54.047 KB	48 B	391 B
2-byte array (short[], char[])	13,946	871.414 KB	24 B	28.023 KB	48 B	61 B

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

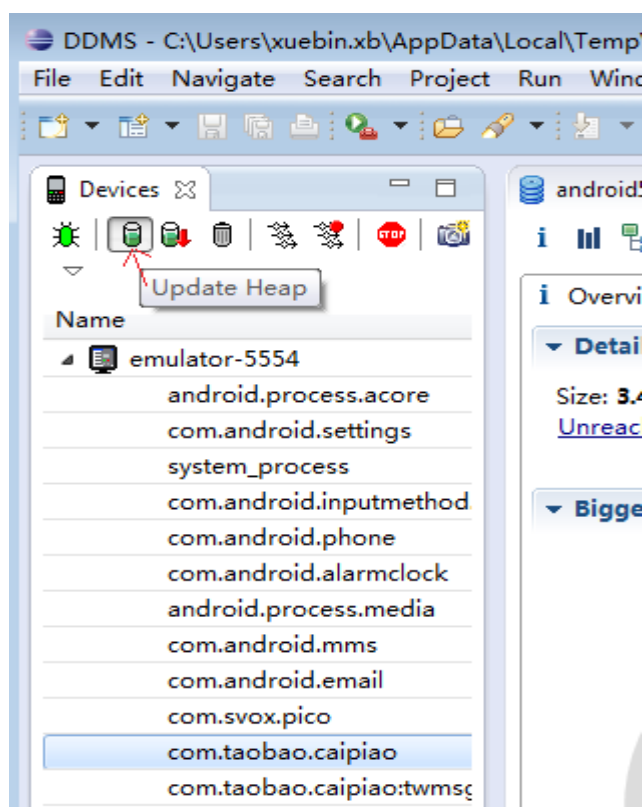
如果在操作中，尤其是循环操作中，发现这两个数的变换稳定在一个区间内，比如增加到某个值后又变小了，则为正常，如果发现这两个数据持续上升，无回落，则有内存泄露的风险。

黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

b) 获取 dump 文件，分析文件

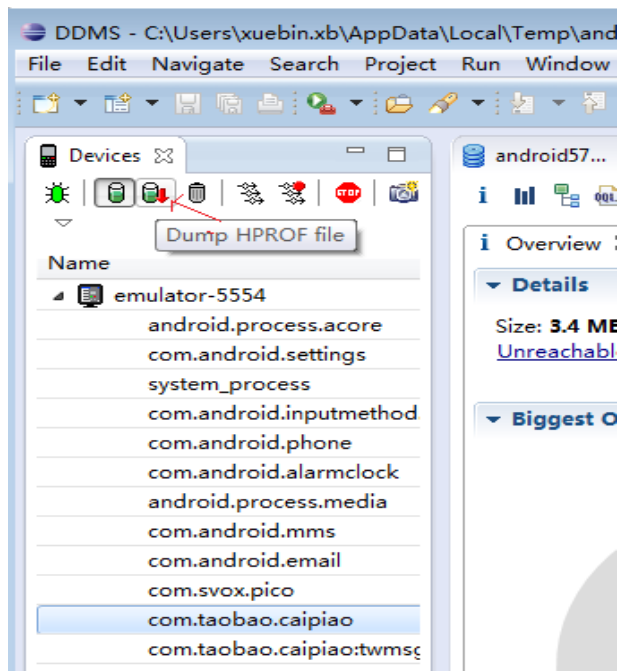
选中要分析的进程，点击 Update heap 图标，如下图所示。



点击 update heap 后，开始操作真机或模拟器，也即是我们要进行的测试，本例是操作彩票客户端的买精彩足球混合玩法彩票，操作完后，点击 Dump Hprof file 图标，如下图所示。

黑冰老师

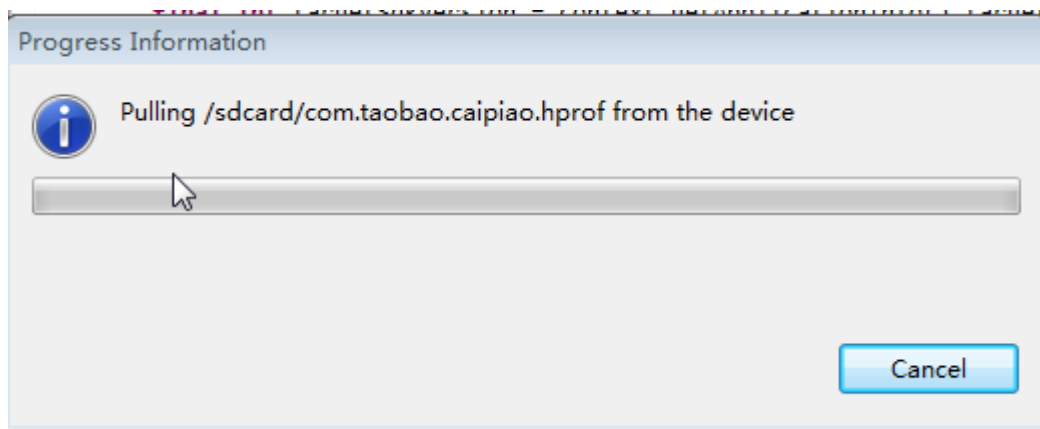
QQ: 1 2 3 5 1 2 8 9 7



黑冰老师

QQ: 123512897

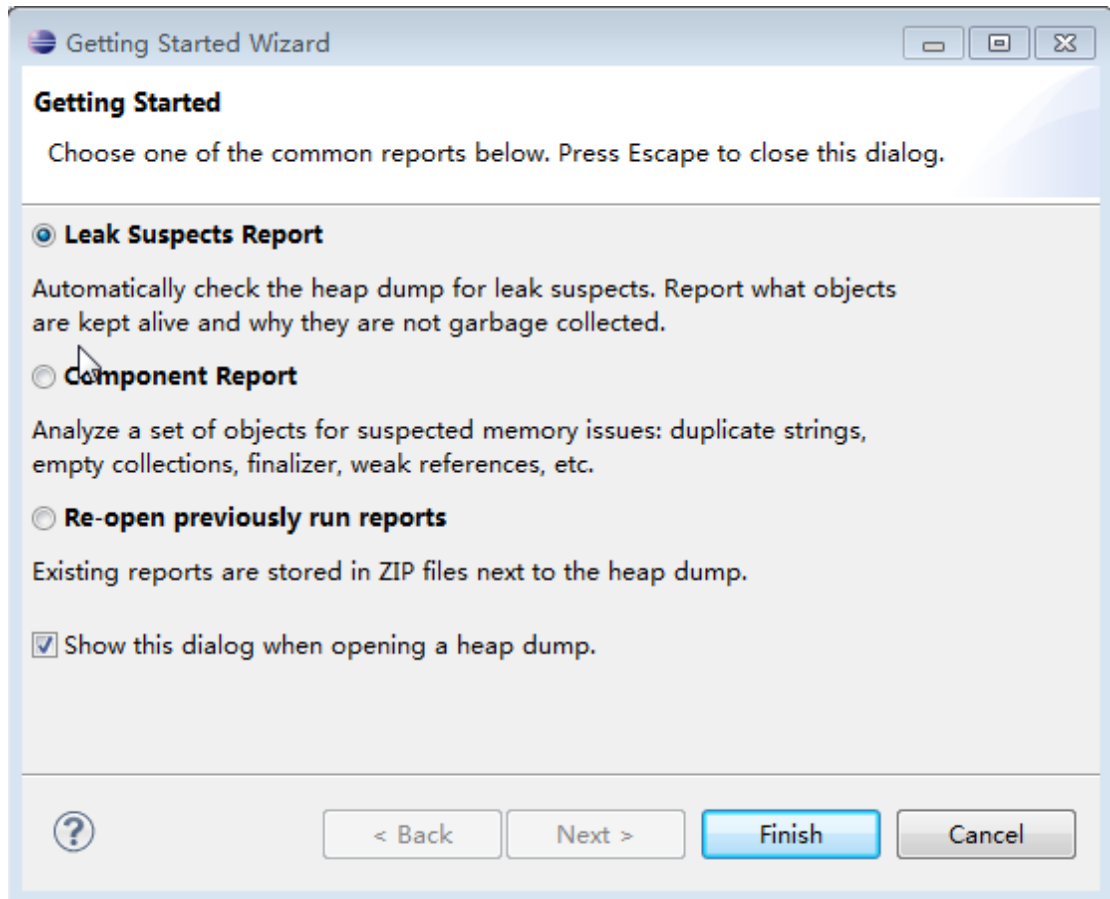
然后会弹出提示框（默认的吧 dump 文件生成在 sdcard/目录下），生成文件需要大约五六分钟。



然后会弹出图下图所示框，确保“Leak Suspects report”选中状态，点“finish”。

黑冰老师

QQ: 123512897



然后就会出现下图所示的结果，有各种内存信息。

黑冰老师

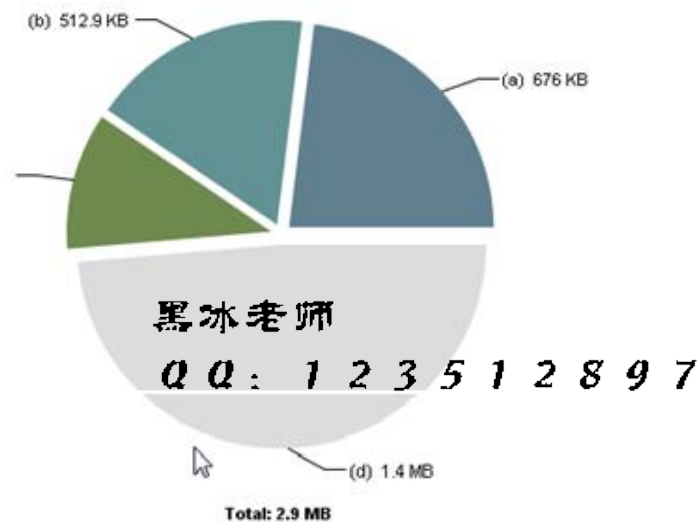
QQ: 1 2 3 5 1 2 8 9 7

Leak Suspects

System Overview

Leaks

Overview



Problem Suspect 1

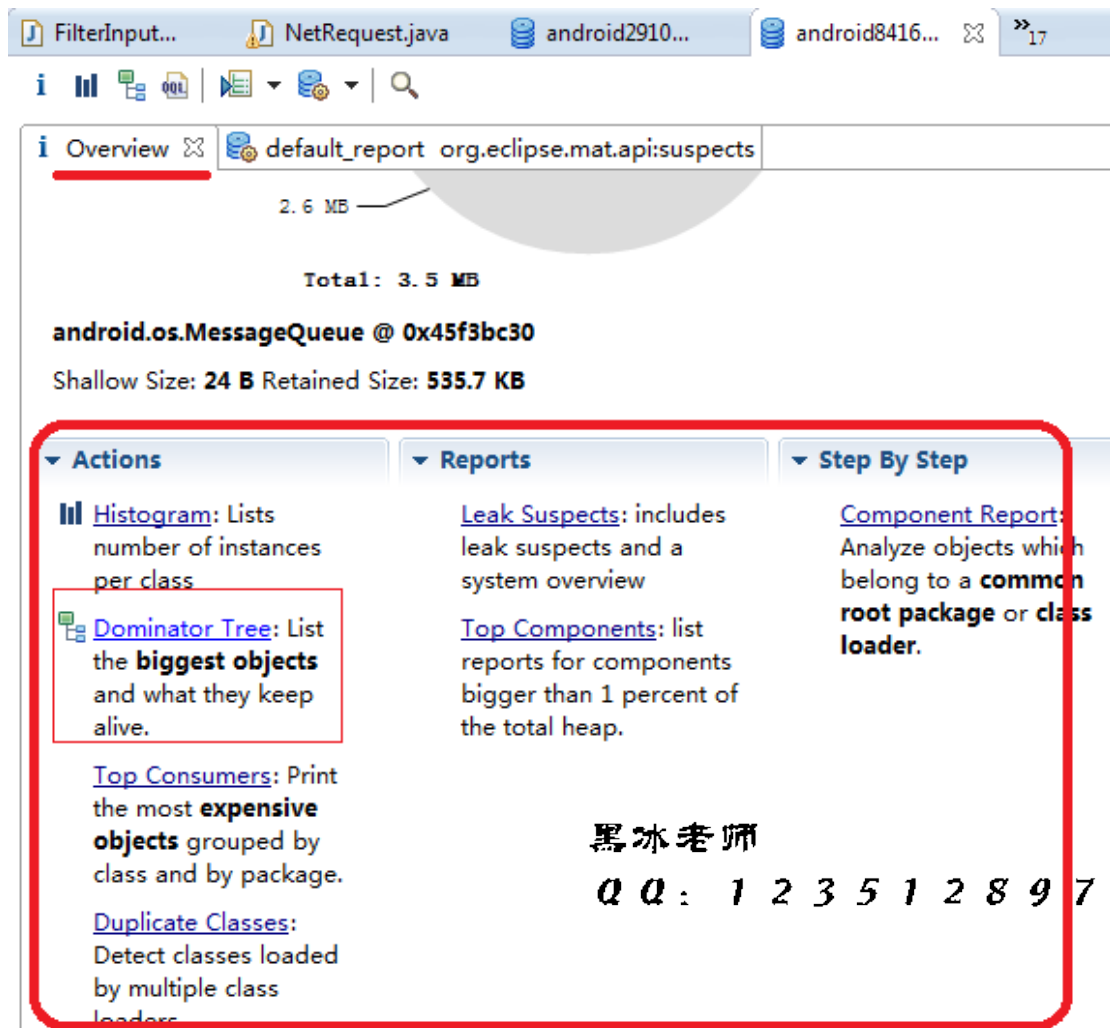
3,286 instances of "java.lang.Class", loaded by "<system class loader>" occupy **692,216 (23.04%)** bytes.

Biggest instances:

- class com.ibm.icu4jni.util.Resources\$DefaultTimeZones @ 0x40161c30 - 165,488 (5.51%) bytes.
- class android.text.Html\$HtmlParser @ 0x400ef1f8 - 126,592 (4.21%) bytes.
- class org.apache.harmony.security.fortress.Services @ 0x40071508 - 51,456 (1.71%) bytes.
- class android.content.res.Resources @ 0x4004dfe8 - 35,608 (1.19%) bytes.
- class android.text.AutoText @ 0x400f2388 - 31,344 (1.04%) bytes.

黑冰老师

QQ: 123512897



4.插件是生成的 hprof 文件，默认实在 AppData\local\temp**下，也可以直接打开该文件。如果打不开的话，把该文件拷贝到 Androidsdk\tool 目录下，最新版不需要再做转换，老版本的需要做转换后才能在 M A T 中打开。

装换方法如下：

运行 cmd 打开命令行，cd 到\ android-sdk-windows\tools 所在目录，并输入命令 hprof-conv xxxxx.hprof yyyyy.hprof，其中 xxxxx.hprof 为原始文件，yyyyy.hprof 为转换过后的文件。转换过后的文件自动放在 android-sdk-windows\tools 目录下。

5.打开 MAT：

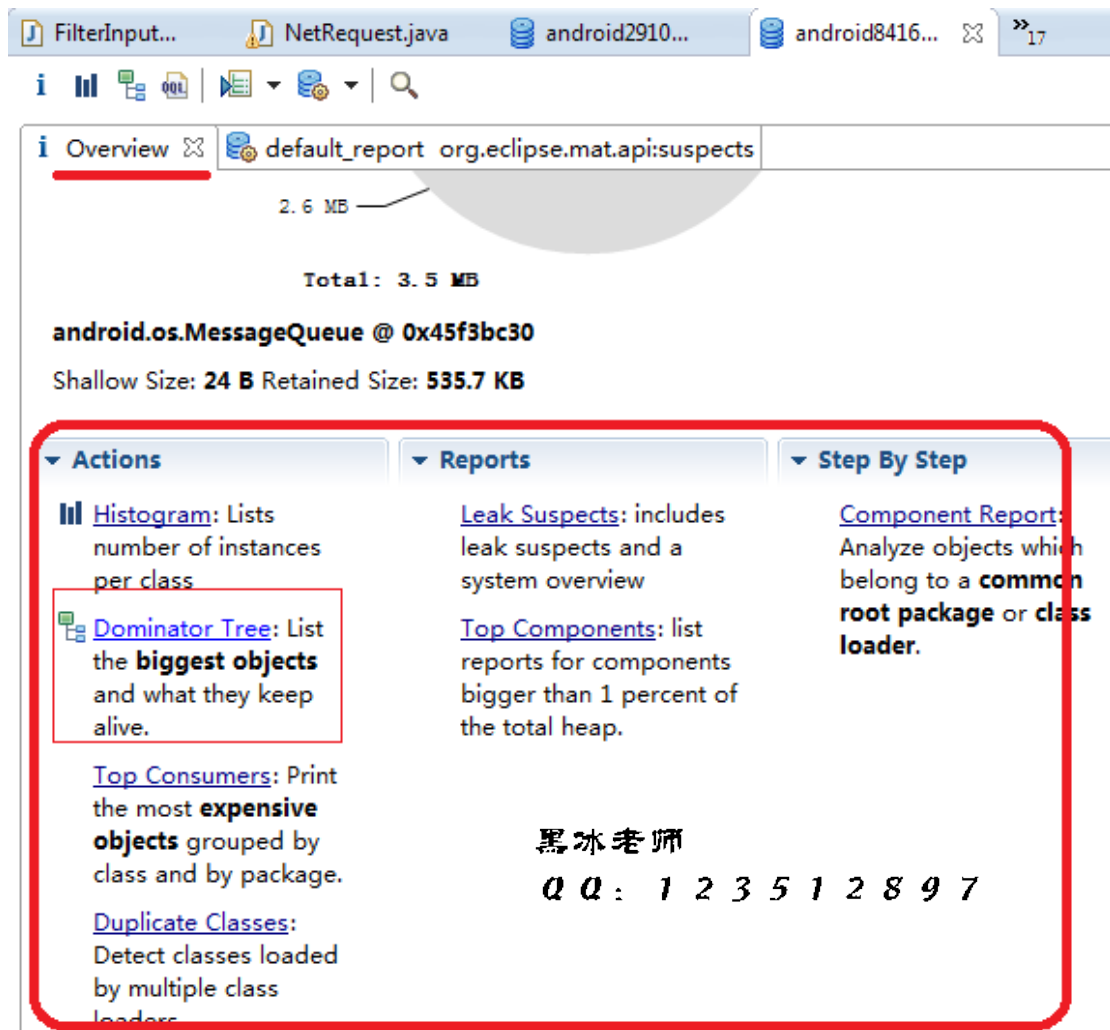
在 Eclipse 中点击 Windows->Open Perspective->Other->Memory Analysis

6.导入.hprof 文件（导入.hprod 的操作也适用于在 DDMS 中直接操作查看）

在 MAT 中点击 File->Open File,浏览到刚刚转换而得到的.hprof 文件，并 Cancel 掉自动生成报告。

（以下操作也适用于直接在 DDMS 中查看 hprof 文件）

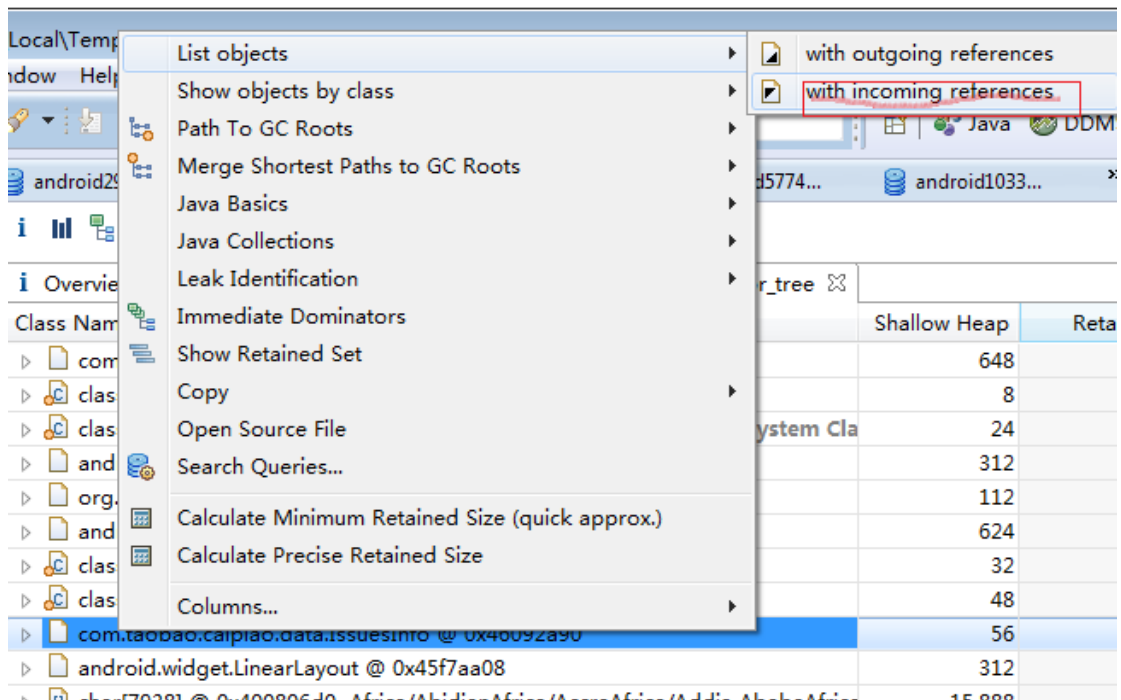
点击 Dominator Tree，并按 Package 分组。



选择自己所定义的 Package 类点右键，在弹出菜单中选择 List objects->With incoming references，如下图所示。

黑冰老师

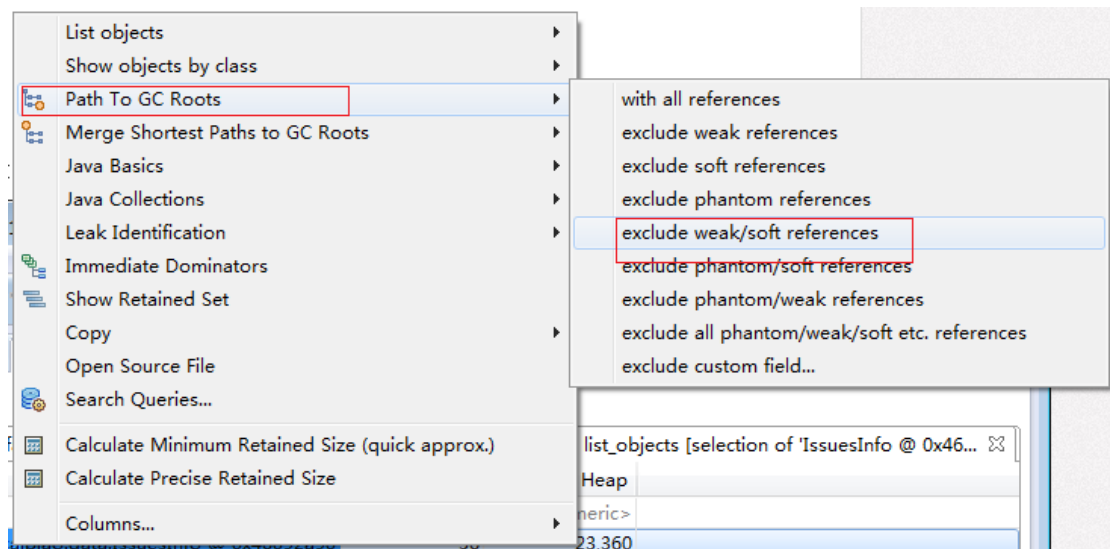
QQ: 1 2 3 5 1 2 8 9 7



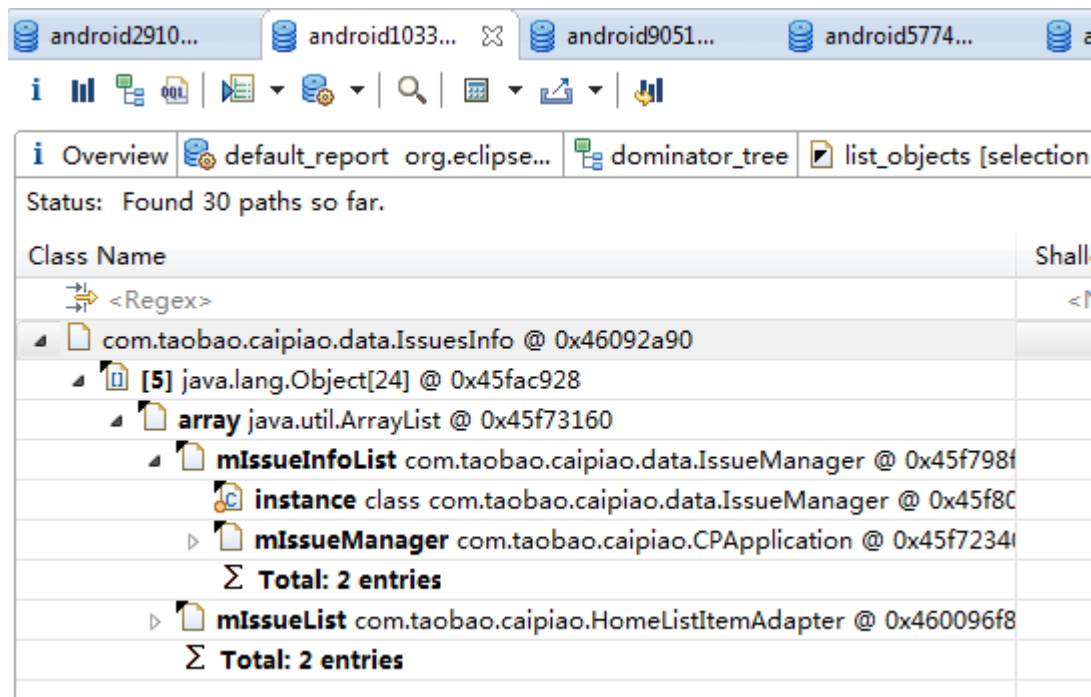
黑冰老师

QQ: 1 2 3 5 1 2 8 9 7

这时会列出所有可疑类，右键点击某一项，并选择 Path to GC Roots->exclude weak/soft references。



会进一步筛选出跟程序相关的所有有内存泄露的类。据此，可以追踪到代码中的某一个产生泄露的类，以彩票为例子，选了某一个包的结果如下。



[i Overview](#)
[default_report org.eclipse...](#)
[dominator_tree](#)
[list_objects \[selection\]](#)

Status: Found 30 paths so far.

Class Name	Shall
<Regex>	<!
com.taobao.caipiao.data.IssuesInfo @ 0x46092a90	
[5] java.lang.Object[24] @ 0x45fac928	
array java.util.ArrayList @ 0x45f73160	
mIssueInfoList com.taobao.caipiao.data.IssueManager @ 0x45f798f	
instance class com.taobao.caipiao.data.IssueManager @ 0x45f8C	
mIssueManager com.taobao.caipiao.CPApplication @ 0x45f7234	
Σ Total: 2 entries	
mIssueList com.taobao.caipiao.HomeListItemAdapter @ 0x460096f8	
Σ Total: 2 entries	

黑冰老师

Q Q : 1 2 3 5 1 2 8 9 7