

# JProfiler 入门教程

---

V1.1 适用于 JProfiler 7.1.2



作者：赵磊

博客：<http://elf8848.iteye.com/>

2012/8/8

# 目录

一、JProfiler 介绍 .....	4
二、启动 JProfiler 7 .....	4
2.1 Quick Start 菜单 .....	4
2.2 Start Center 菜单 .....	5
2.2.1 Start Center-> Open Session 菜单介绍 .....	5
2.2.2 使用 JProfiler 自带的示例演示内存泄露和线程阻塞 .....	5
2.2.3 Start Center-> New Session 菜单介绍 .....	14
三、JProfiler 的监控方式介绍: .....	14
1. 等待模式 .....	14
2. 非等待模式 .....	14
3. 离线模式 .....	14
四、JVM 工具接口(JVM TI)介绍 .....	15
五、服务端 JProfiler 代理工具(Agent) 加载原理 .....	15
六、创建监视本地 Tomcat 的工程（等待模式） .....	15
6.1 进入创建菜单 .....	16
6.2 选择使用的服务器容器 .....	16
6.3 选择 Tomcat 容器的位置 .....	17
6.4 选择虚拟机类型 .....	18
6.5 确认启动监控的方式 .....	19
6.6 选择容器启动脚本 .....	20
6.7 选择监控端口 .....	21
6.8 最后确认信息 .....	22
6.9 启动容器 .....	23
七、创建监视远程 Tomcat 的工程（等待模式） .....	24
7.1 测试环境 .....	24
7.2 JProfiler 软件下载地址 .....	24
7.3 客户端 JProfiler 安装 .....	24
7.4 服务器端 JProfiler 安装 .....	24
7.5 客户端连接配置 .....	25

---

八、创建监视远程 Tomcat 的工程（非等待模式） .....	26
九、创建监视远程 Java 程序的工程.....	26
十、其它内容 .....	27
10.0 局域内使用同一个 license key 问题.....	27
10.1 JProfiler 对程序性能的影响 .....	28

## 一、JProfiler 介绍

JProfiler 是一个非常优秀的 JVM 分析工具，可监视本地和远程的 JVM，适用于各种操作系统。常用的功能有：

- 1、监视堆内存占用情况和创建对象实例的数量找出内存泄露。
- 2、监视占用 CPU 较多的方法
- 3、监视线程的阻塞与死锁
- 4、监视 GC 的耗时。

监视本地 JVM (Windows):  
需要安装 JProfiler windows 版本

监视远程的 JVM(Linux):  
客户端需要安装 JProfiler windows 版本  
服务需要安装 JProfiler Linux 版本  
客户端与服务端通过 socket 通信

JProfiler 软件下载地址:  
<http://www.ej-technologies.com/>  
<http://www.ej-technologies.com/download/jprofiler/files.html>  
(不需要注册就可以下载)

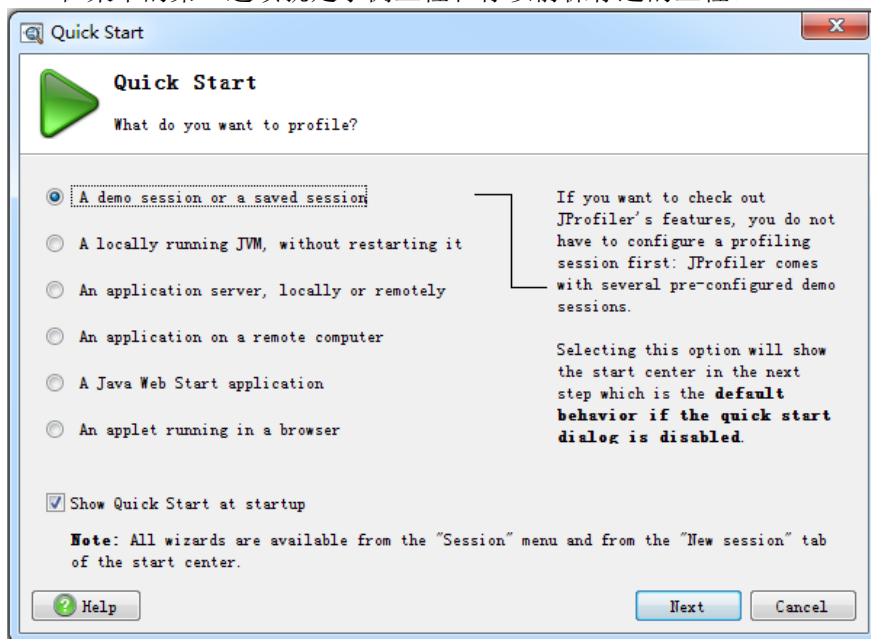
本文以 JProfiler 7.1.2 为例进行讲解。

## 二、启动 JProfiler 7

### 2.1 Quick Start 菜单

每次启动弹出“快速启动”菜单，它的不重要，可以关闭它。“快速启动”菜单能实现的功能在后面的 Start Center 菜单中都可以实现。

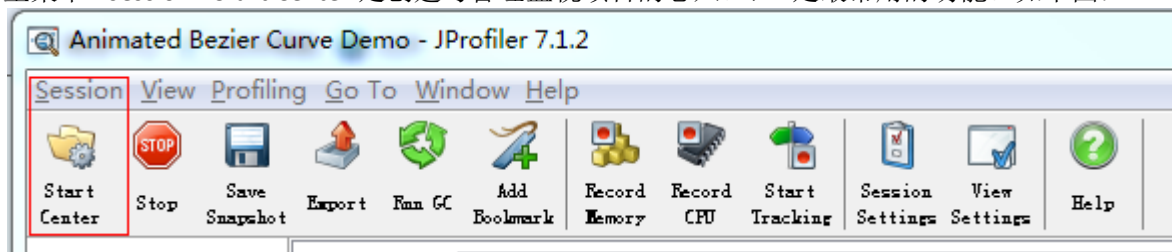
在菜单的第一选项就是示例工程和你以前保存过的工程。



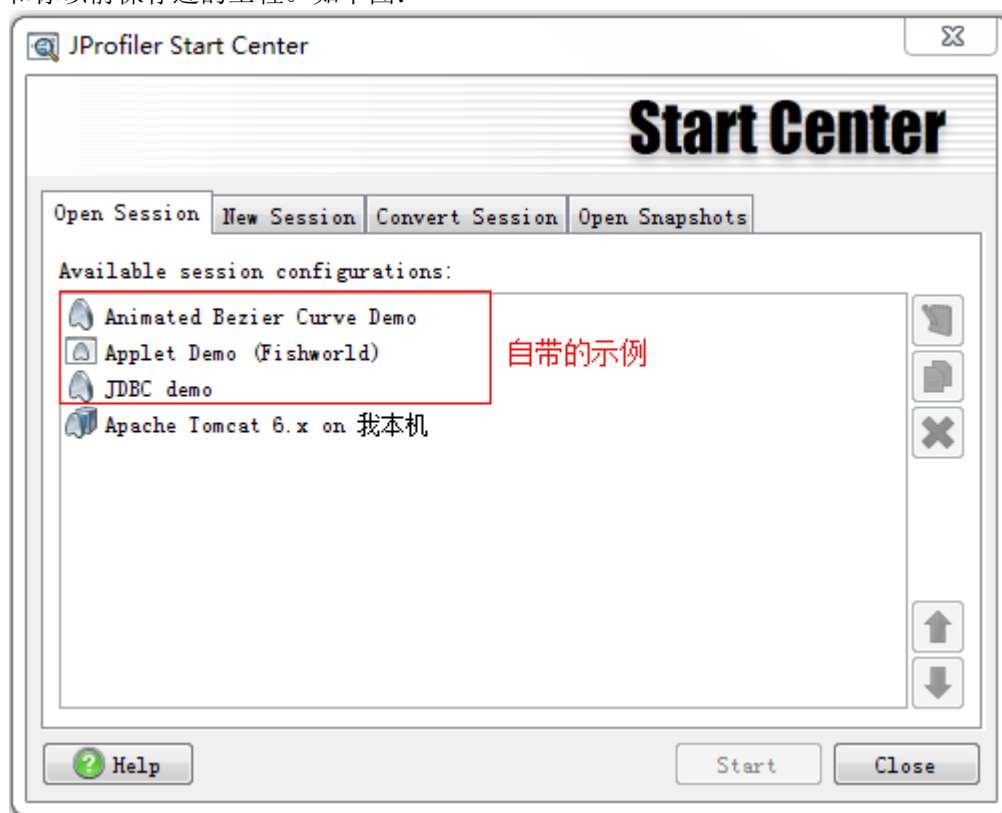
## 2.2 Srart Center 菜单

### 2.2.1 Srart Center-> Open Session 菜单介绍

主菜单->session->Srart Center 是创建与管理监视项目的总入口，是最常用的功能。如下图：



每次打开，默认展示的是第一个选项卡“Open Session”。在菜单的第一选项卡“Open Session”中是示例工程和你以前保存过的工程。如下图：

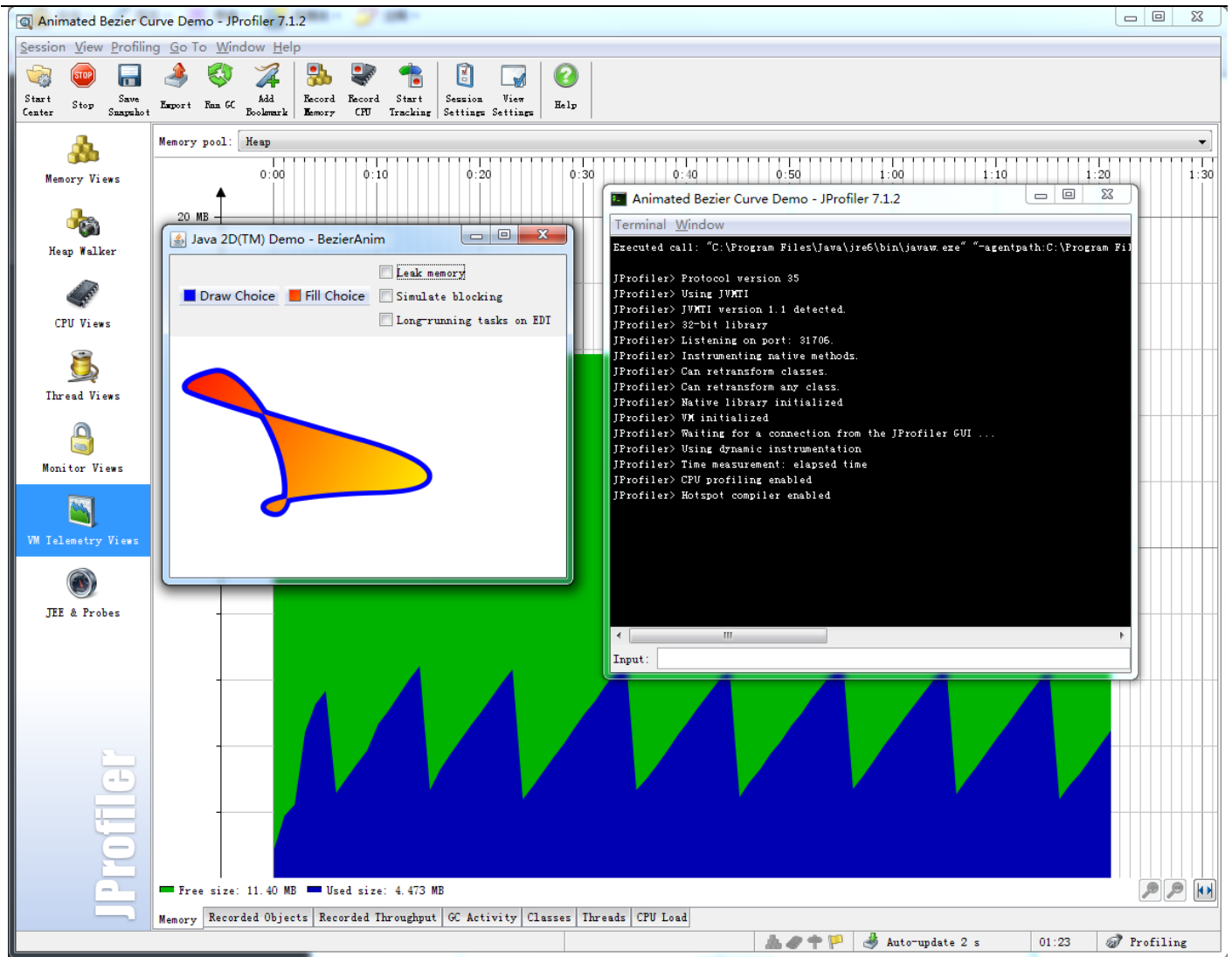


其中 Animated 是一个监视 Swing 动画的演示、JDBC 是一个监视 JDBC 的演示、Applet 是一个监视网页小程序演示。

### 2.2.2 使用 JProfiler 自带的示例演示内存泄露和线程阻塞

我们来演示内存泄露和线程阻塞，主菜单->session->Srart Center-> Open Session 窗口中，选第一个示例项目 Animated Bezier Curve Demo，点击 Start 按钮，在下一个窗口中都使用默认值，按 OK 按钮。这时示例程序就开始运行了，你就可以监视 JVM 的状态了。并且可以手动控制是否开始 模拟内存泄露，和模拟线程阻塞的情况。

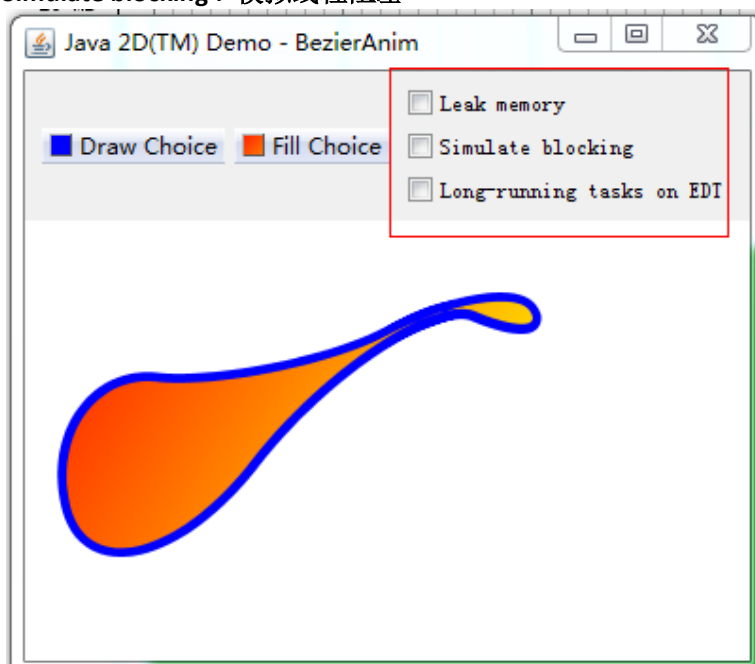
我们就以这个示例，来讲解 JProfiler 的入门使用方法。  
运行起来后情况如下图：



在下图中有前两个复选框，可以模拟内存泄露，线程阻塞的情况。

**Leak memory:** 模拟内存泄露

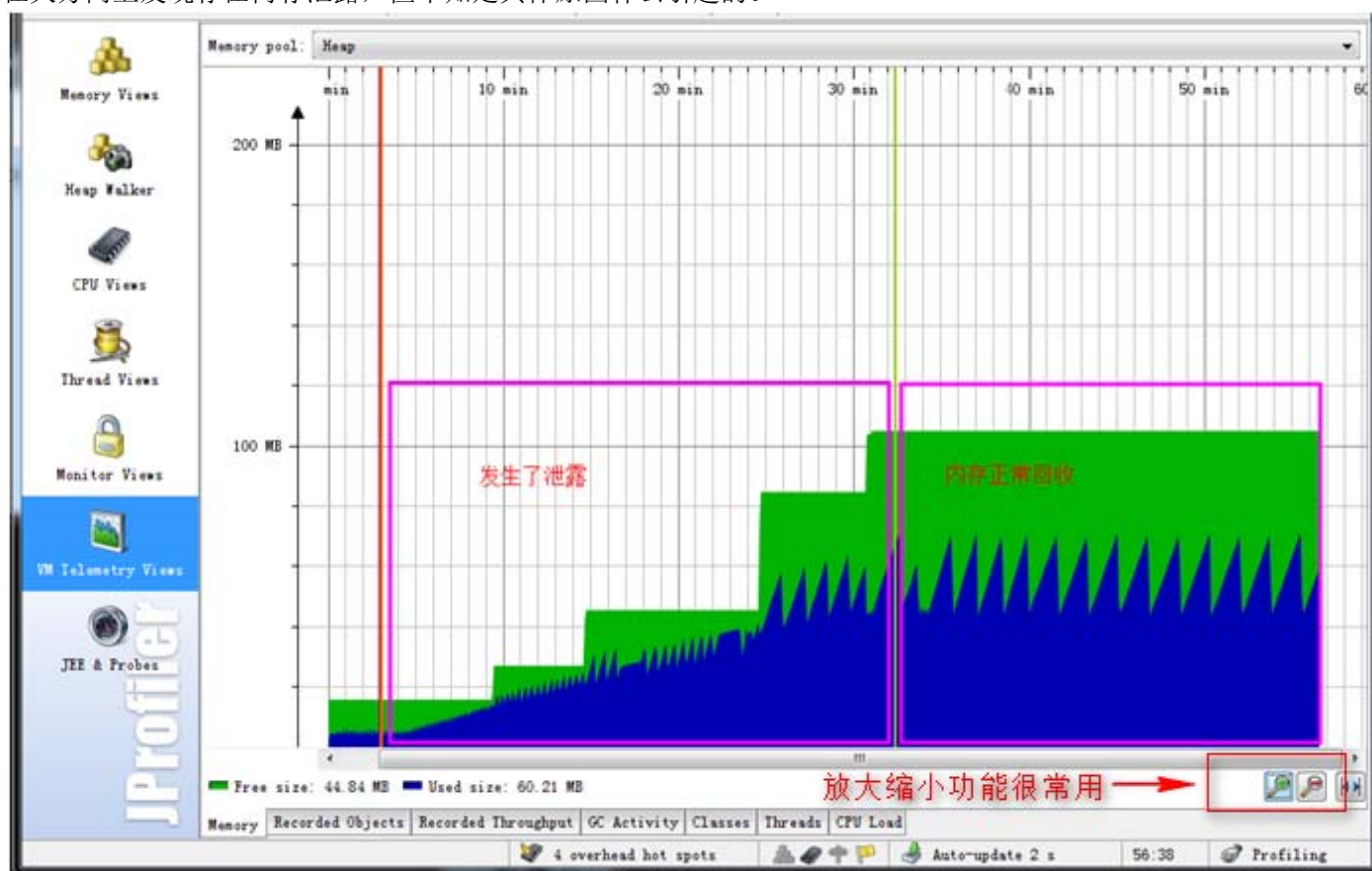
**Simulate blocking:** 模拟线程阻塞



### 2.2.2.1 监视堆内存的增长情况

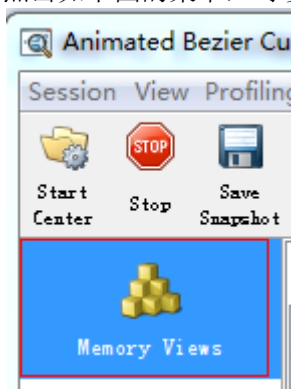
经过长时间的运行，发现如下图的情况：

下图：前半部分是模拟内存泄露，后半部分是关闭了模拟内存泄露。可发现内存泄露时 Heap 内存呈增长趋势。可在大方向上发现存在内存泄露，但不知是具体原因什么引起的。



### 2.2.2.2 查看内存中对象的实例数量

点击如下图的菜单，可以开始查看内存中对象的实例数量

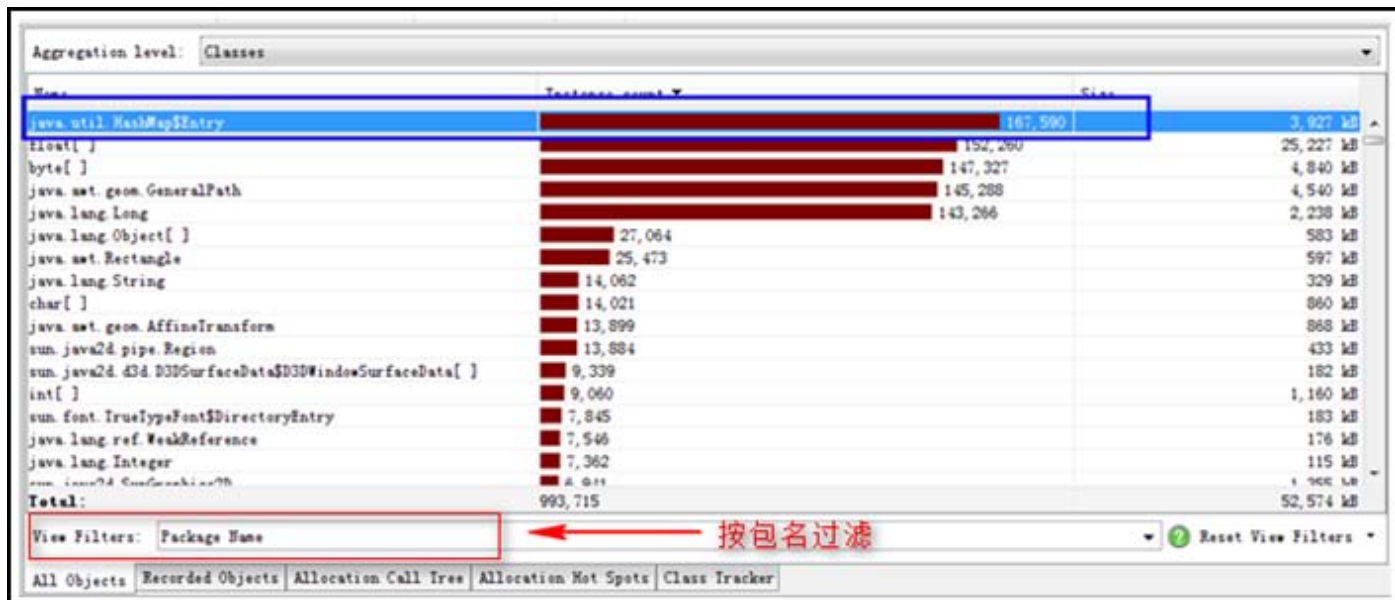


查看内存中对象的实例数量，找出不能被回收的对象。经过长时间的观察，下图中的 HashMap\$Entry 实例数量

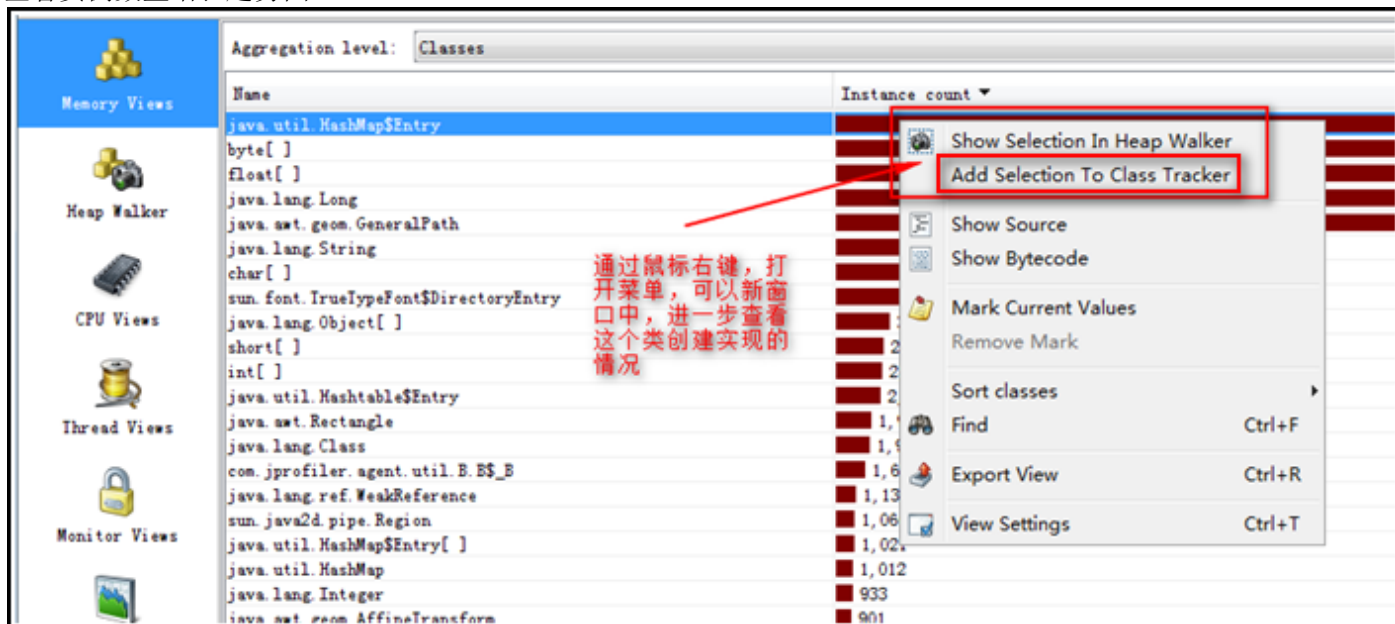
一直在增长，最早时 HashMap\$Entry 数量是 1 万左右，现在是 16 万多。可以断定就是它没有被释放。

不一定数量最多的实例就是泄露，要根据业务情况分析。如果 1 个用户登录操作数据库，创建了一个连接，用户退出后，这个连接实现数量没有-1（你开发时设计的会-1），就证明发生了泄露。

可以通过包名过滤，只显示你开发的类，这个功能是非常有用的。

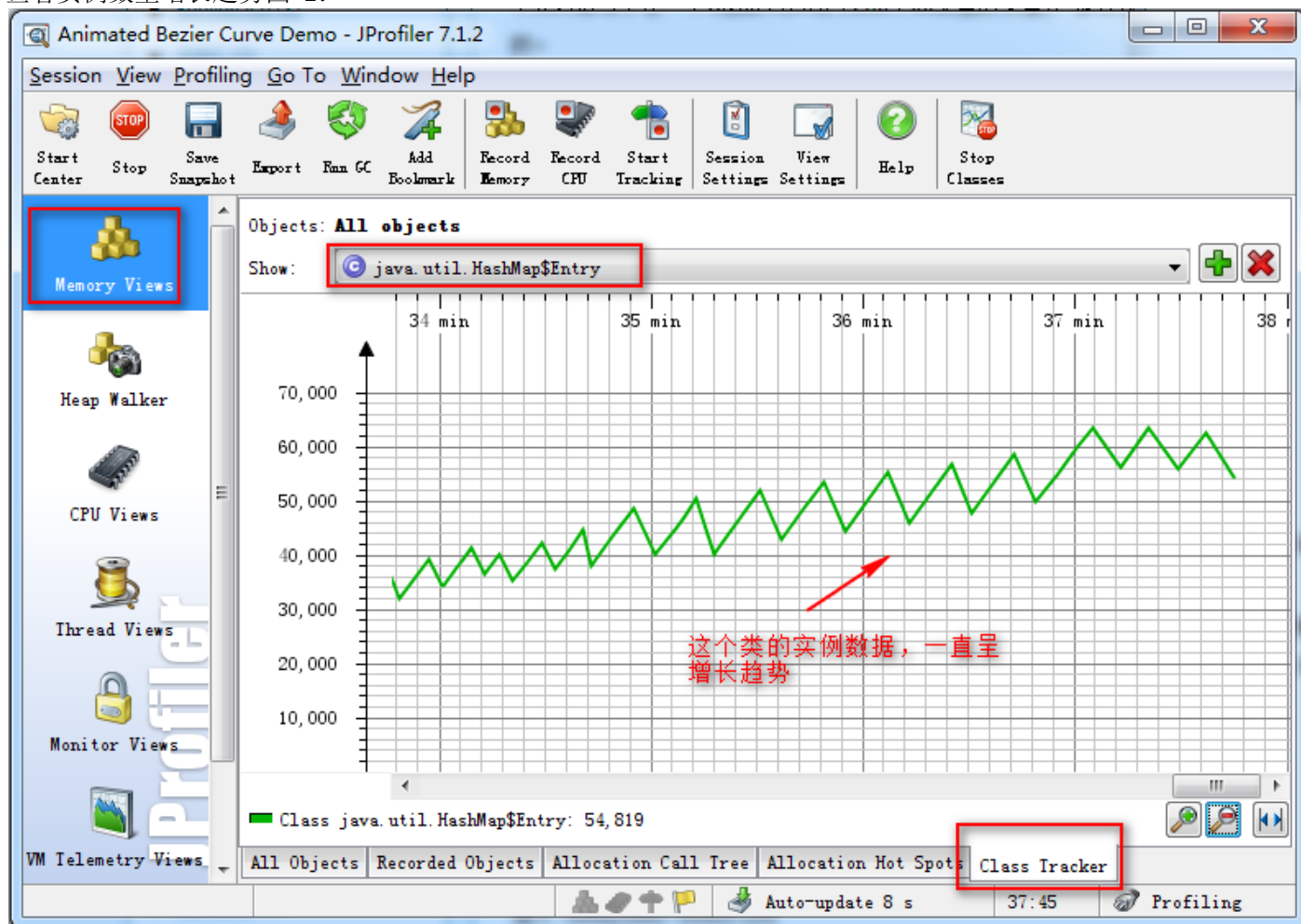


查看实例数量增长趋势图 1:





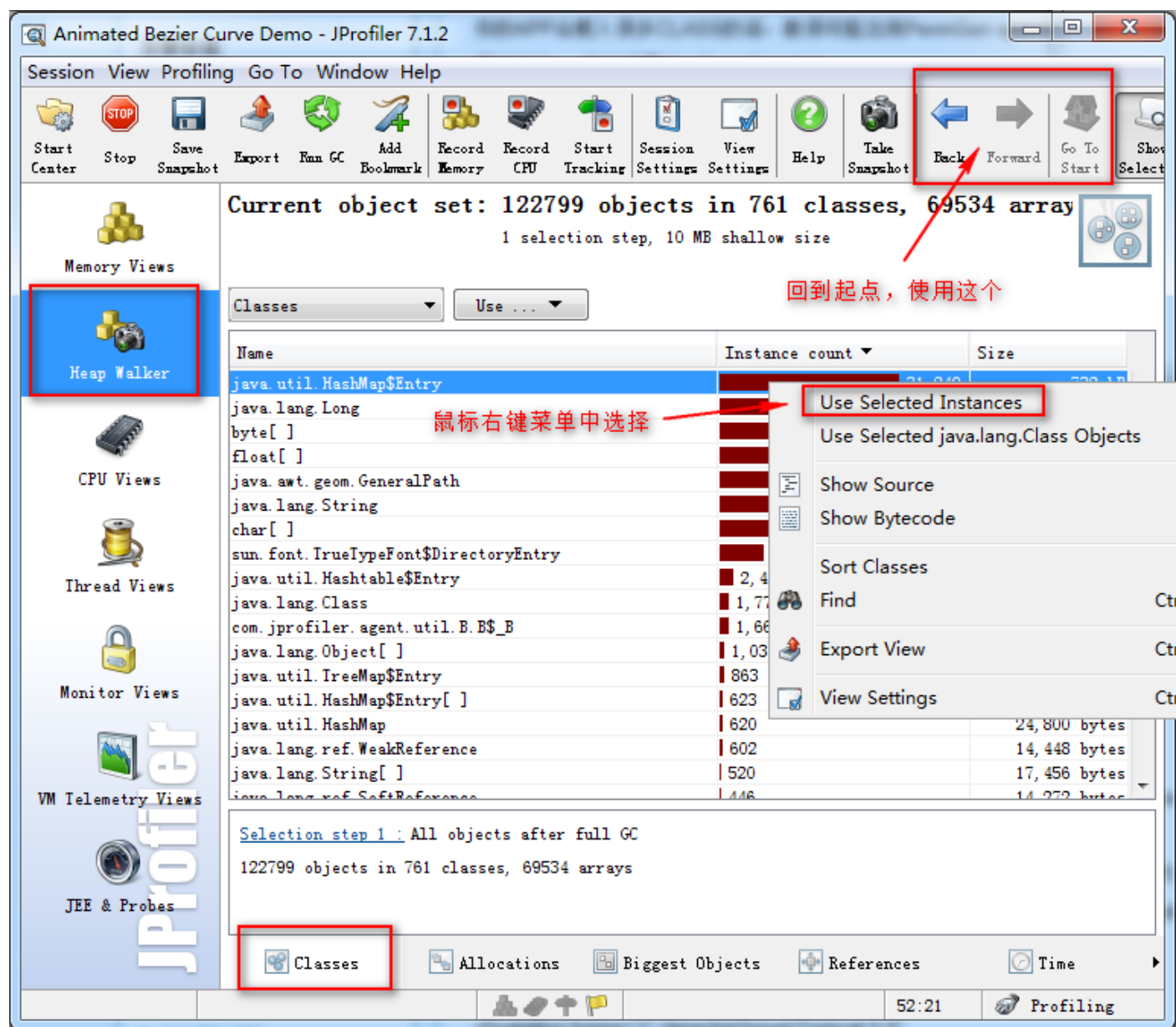
查看实例数量增长趋势图 2:



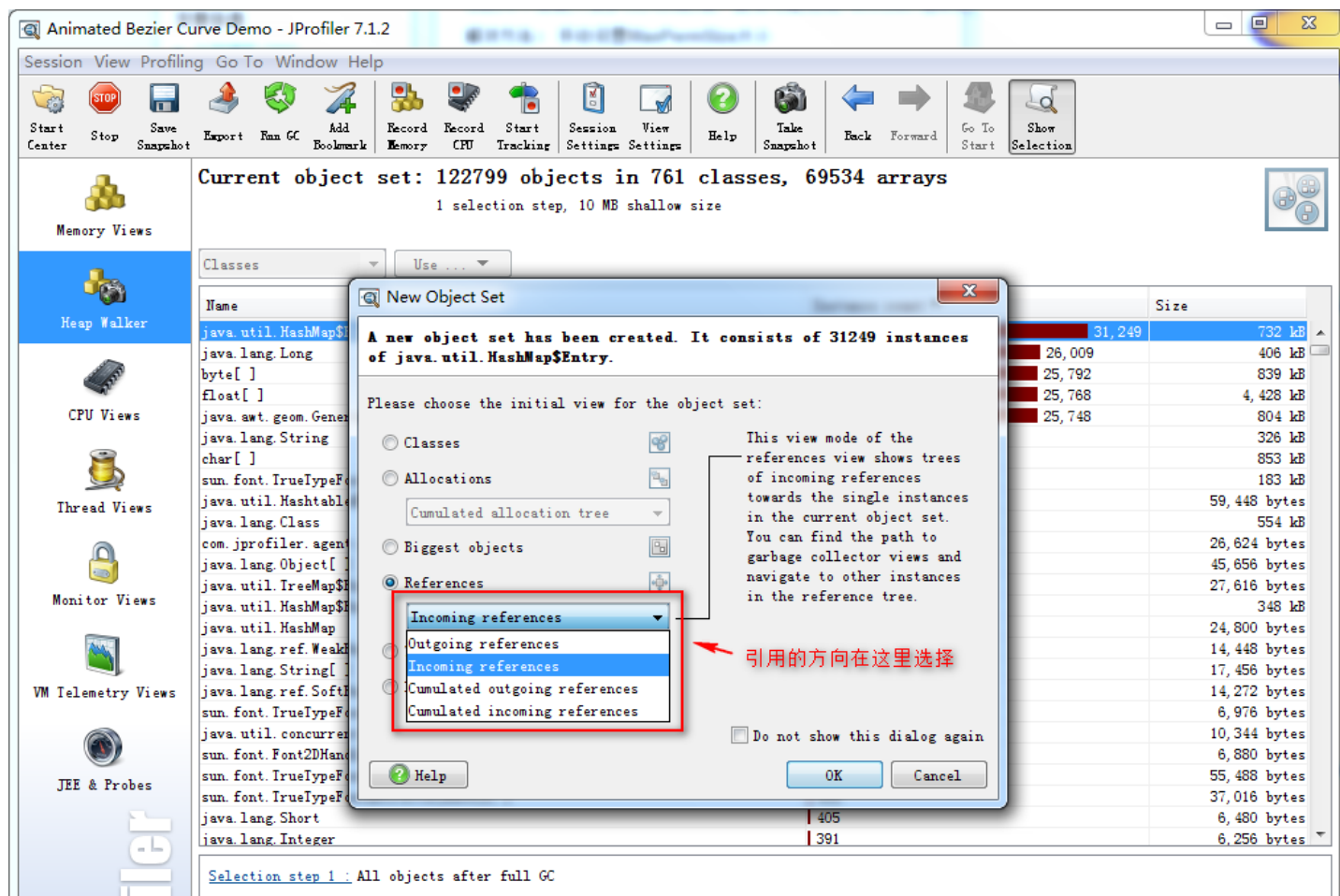
## 2.2.2.3 查看实例被谁引用

在左侧的主菜单，选 Heap Walker，并点相机图标  开始。

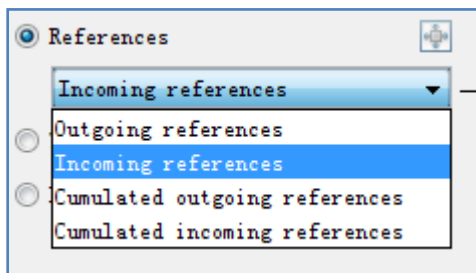
在下图中，可以通过鼠标右键打开快捷菜单，选择 Use Selected Instances



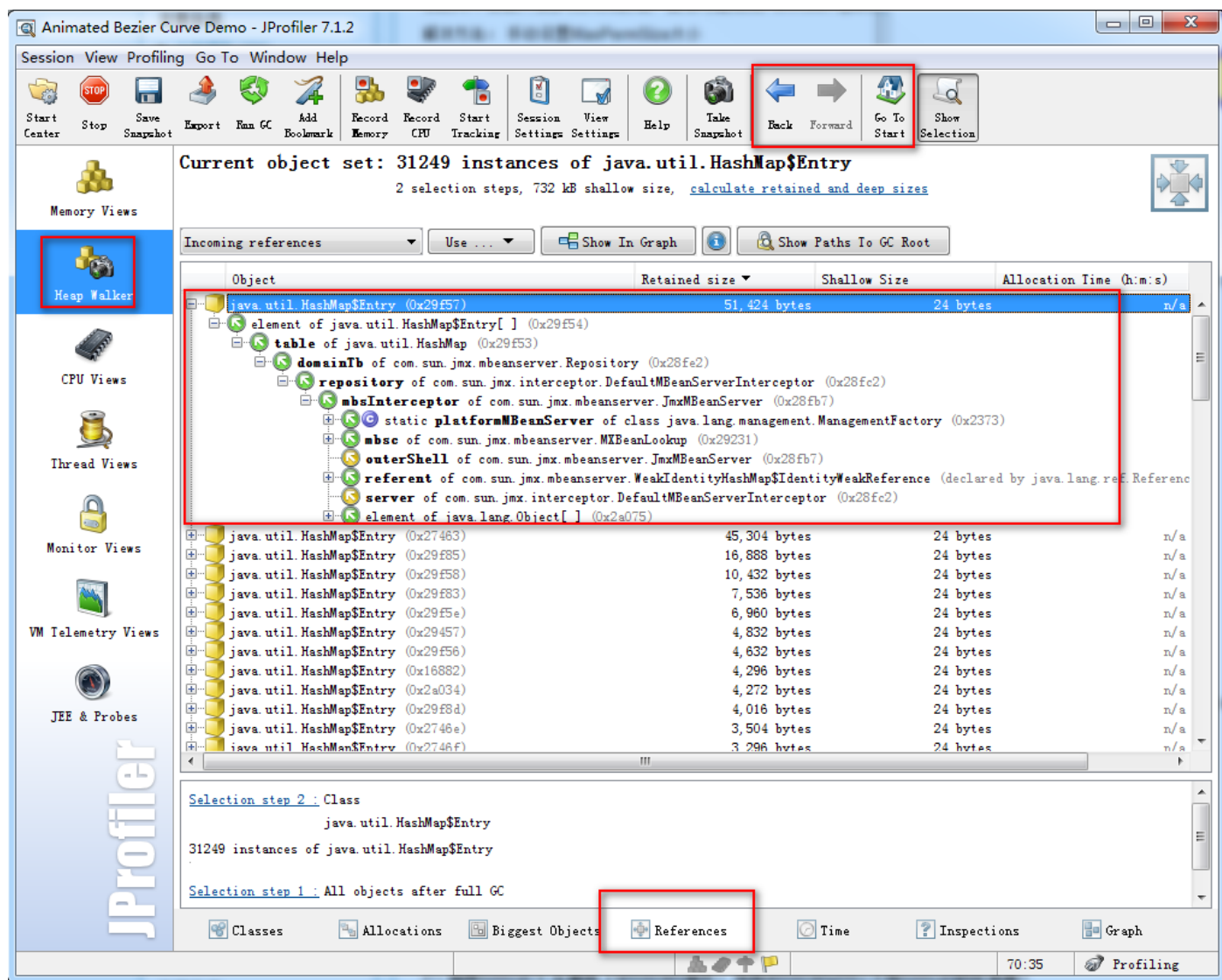
选择引用的方向，见下图：



来一张清晰的图：



当前对象实例被谁引，引用了谁，请看下图：

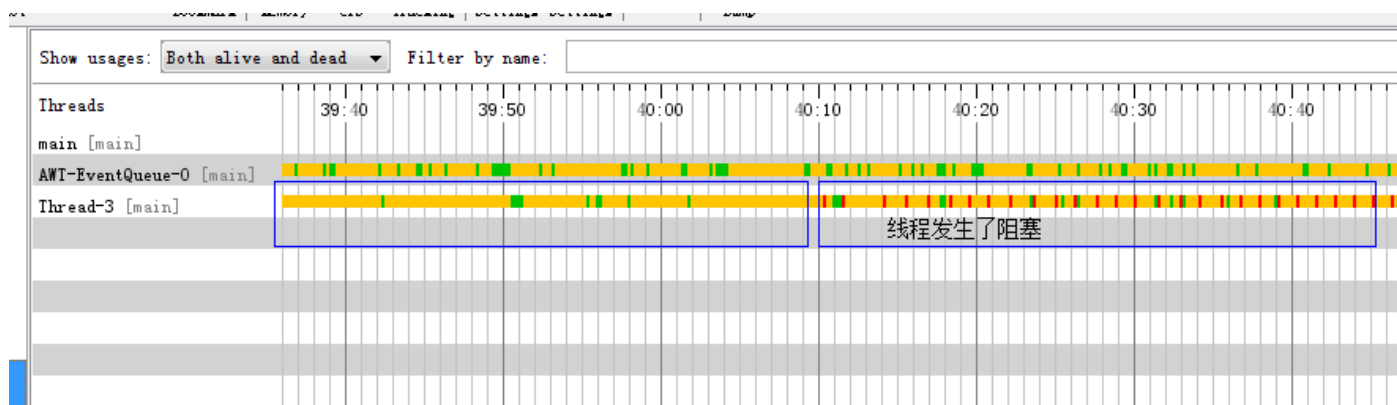


#### 2.2.2.4 监视线程的阻塞情况

下图左边的蓝框中没有红色线段，表明没有发生线程阻塞

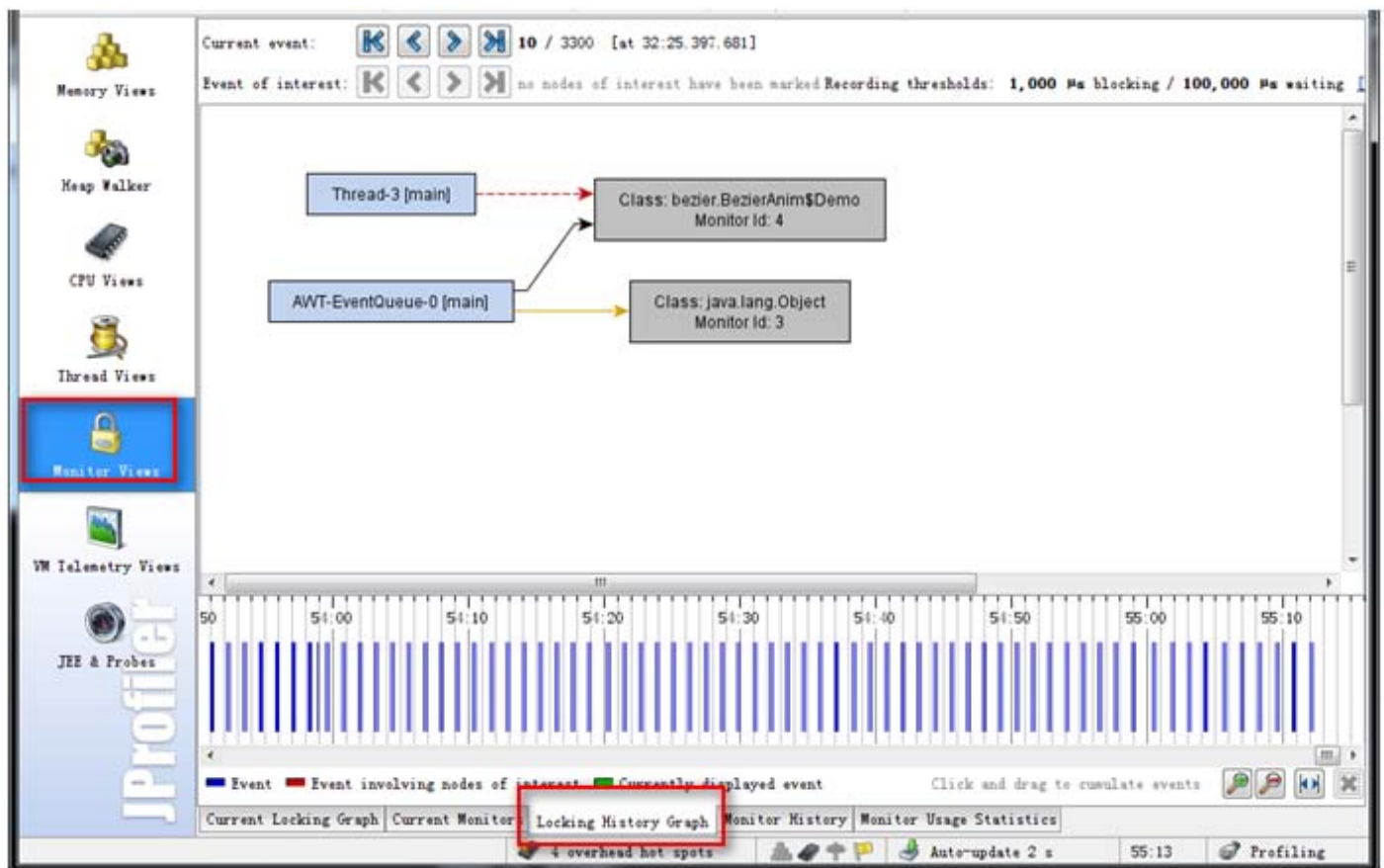
下图右边的蓝框中有红色线段，表明发生了线程阻塞

只知道发生阻塞的线程名，但不知道具体原因。



### 2.2.2.5 找出线程阻塞的点

可以记录历史，一步一步的看，像看慢镜头一样。



到这里，通过 JProfiler 自带的演示程序，已经了解了 JProfiler 的基本使用技巧。看到这里已经算是入门了。

## 2.2.3 Start Center-> New Session 菜单介绍

Start Center-> New Session 选项卡，是最主要最常用的功能。

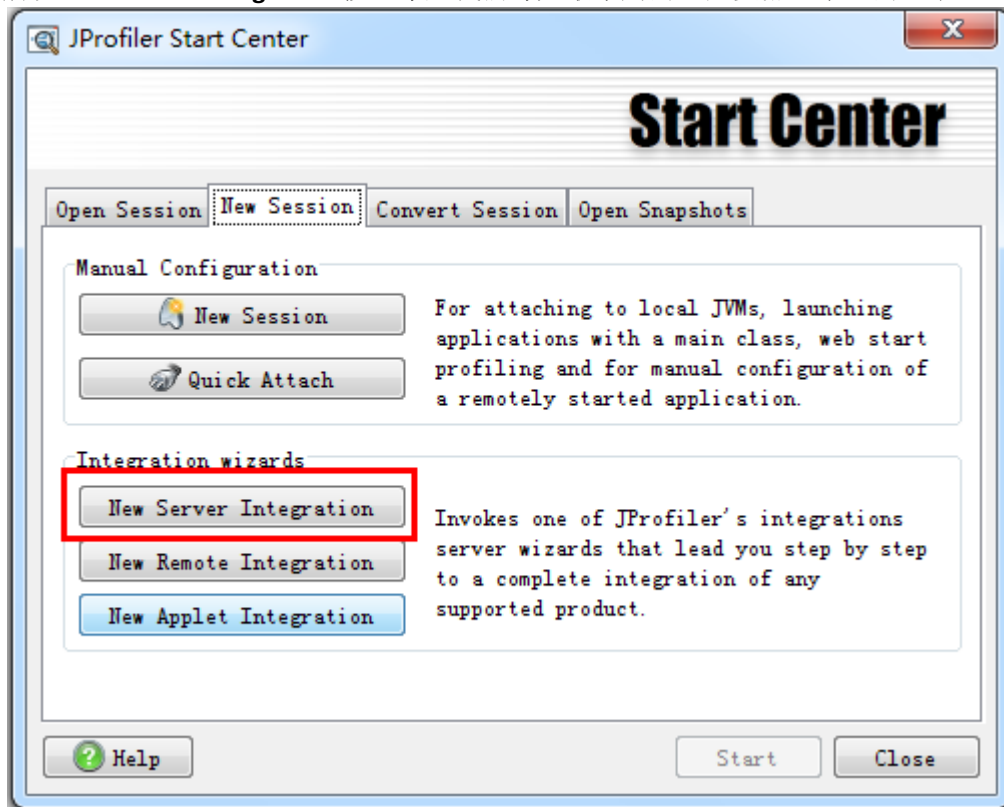
New Session 按钮： 是用于监控 GUI 类型的程序或 Applet 等不需要服务器容器的应用程序

New Server Integration 按钮： 是用于监控服务器容器应用的

New Remote Integration 按钮： 同下

New Applet Integration 按钮： 是对于 New Server Integration 的快捷键，他们直接跳过了使用 New Server Integration 的前几个步骤，使用了默认值。

所以 **New Server Integration** 按钮才是我们每天最常用的。只要熟悉了它的，就 OK 了。后面主要讲它。



## 三、JProfiler 的监控方式介绍：

使用 JProfiler 监视 JVM 时，对连接的时机控制有以下几种：

### 1. 等待模式

wait for a connection from the Jprofiler GUI,此模式为，在启动容器（Weblogic/Tomcat）时，需要等 Jprofiler 连接后才能启动；

JVM TI 是采用事件通知方式告知 JProfiler 相关的 Jvm 的状态变化，等待模式可以不漏掉通知。

### 2. 非等待模式

Start Immediately，此模式下，容器（Weblogic/Tomcat）独立启动，Jprofiler 随时可以连接；

### 3. 离线模式

Profile offline, Jprofiler GUI cannot connect 此模式为离线模式，生成相关记录文件事后分析；



## 四、JVM 工具接口(JVMTI)介绍

Java 虚拟机工具接口 (Java Virtual Machine Tool Interface, JVMTI) 提供了一种编程接口(通过 JNI), 允许软件开发人员创建软件代理程序, 用于监视和控制 Java 应用程序。JVMTI 是 JDK 1.5 中的一种新增功能。它取代了 Java Virtual Machine Profiling Interface (JVMPI)(从 JDK 1.1 起作为一种实验功能)。JVMTI 是 JVMPI 的改进的产物, JVMPI 在不久的将来将被废止。

使用 JVMTI 创建的性能分析工具, 称作代理(Agent)

## 五、服务端 JProfiler 代理工具(Agent) 加载原理

理解加载原理, 对于以后的配置很的帮助。

JProfiler 的服务端分析代理工具, 必须在 JVM 启动的阶段通过 JVM 参数来加载

JDK 1.4 以前使用 JVMPI, JDK 1.5 以后使用 JVMTI

参数如下:

Java <=1.4.2(JVMPI)    -Xrunjprofiler

Java >=1.5.0(JVMTI)    -agentlib:jprofilerti

"-Xrun" 或 "-agentlib:" 他们告诉 JVM, 加载 JVMPI/JVMTI 分析代理工具, 后面的字符串取决于你使用的 JProfiler 库文件的文件名:

windows:            jprofilerti.dll

Linux/unix:        libjprofilerti.so

通过加 (冒号-JVMTI),(等号-JVMTI), 参数可以传递给分析代理,例如:

-Xrunjprofiler:port=10000 或 -agentlib:jprofilerti=port=10000,

参数 port=10000 将被传递给分析代理。

如果 JVM 不能加载指定的本地库, 将退出, 并报告错误消息。

Window 环境 JVM 启动参数示例: (目录是 JProfiler 的安装目录)

-agentpath:C:\PROGRA~1\JPROFI~1\bin\windows\jprofilerti.dll=port=8849

Linux 环境 JVM 启动参数示例: (目录是 JProfiler 的安装目录)

-agentpath:/opt/jprofiler7/bin/linux-x86/libjprofilerti.so=port=8849

以上参数从哪里取得?

当你使用 JProfiler 配置一个监视项目时, 在最后一个窗口, 给依据你输入的各种环境条件件, 自动生成并给出参数。

可以参看: [九、创建监视远程 Java 程序的工程](#) 中和截图

## 六、创建监视本地 Tomcat 的工程 (等待模式)

**测试环境:** 客户端与服务端在一台机器上 : Windows 7, JProfiler 7.1.2 for windows (安装包: jprofiler\_windows\_7\_1\_2.exe)

监视本地的 Tomcat, 看似是本地, 其实 JProfiler GUI 在一个单独的 JVM 里启动, 他与被监视的目标 jvm 之间通过 socket 通讯, 目的是为了不干扰目标 JVM。

所以监视本地 Tomcat 与监视远程的 Tomcat 的配置方法基本是一样的。你会了当前这个配置, 再配置监视远程的 Tomcat 时就很容易了。

等待模式与非等待模式也只有一点区别。等待模式生成的 jvm 启动参数是

-agentpath:/opt/jprofiler7/bin/linux-x86/libjprofilerti.so=port=8849, 非等待模式生成的 jvm 启动参数是

-agentpath:/opt/jprofiler7/bin/linux-x86/libjprofilerti.so=port=8849,nowait。只是最后面多了一个 nowait。你会了当前这个配置, 再配置非等待模式也是非常容易了。

## 基本思路:

JProfiler 在 Tomcat 的 bin 目录下, 新生成的 startup\_jprofiler.bat 文件, 其中添加了上面提到的参数, 可以用来启动 Tomcat。就可以接受 JProfiler 的连接, 开始监视了。

## 6.1 进入创建菜单

主菜单: session->Start Center 打开 Start Center 菜单。

选择 “New Session” 选择卡,

选择 “New Server Integration” 按钮。

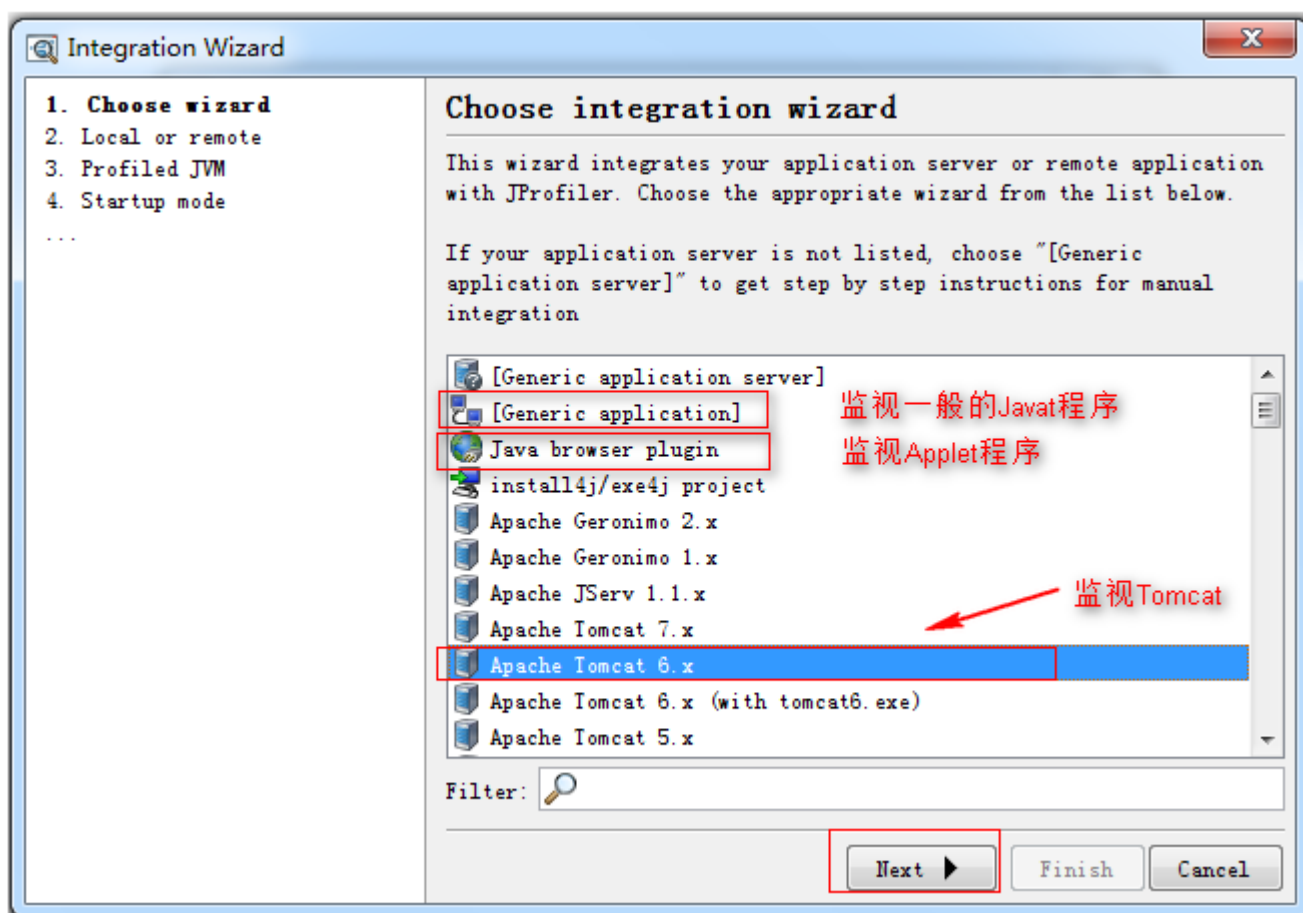
## 6.2 选择使用的服务器容器

在面板中显示出了所有支持可以监视的多种服务器类型, 包括 Tomcat, WebSphere 等。

同时还看到 Generic application, 是监视一般的 Java 程序的。因为 Tomcat 本身也是 Java 程序, 所以选这个也好使。

这里我还是选用 Tomcat6.0, Jprofiler 会生成 Tomcat 的启动文件 (其中加入了相关的 Jvm 启动参数), 对 “等待模式” 的支持更好一些。比手动加 Jvm 启动参数方便一些, 但本质都是一样的。

(不建议使用 with tomcat6.exe )



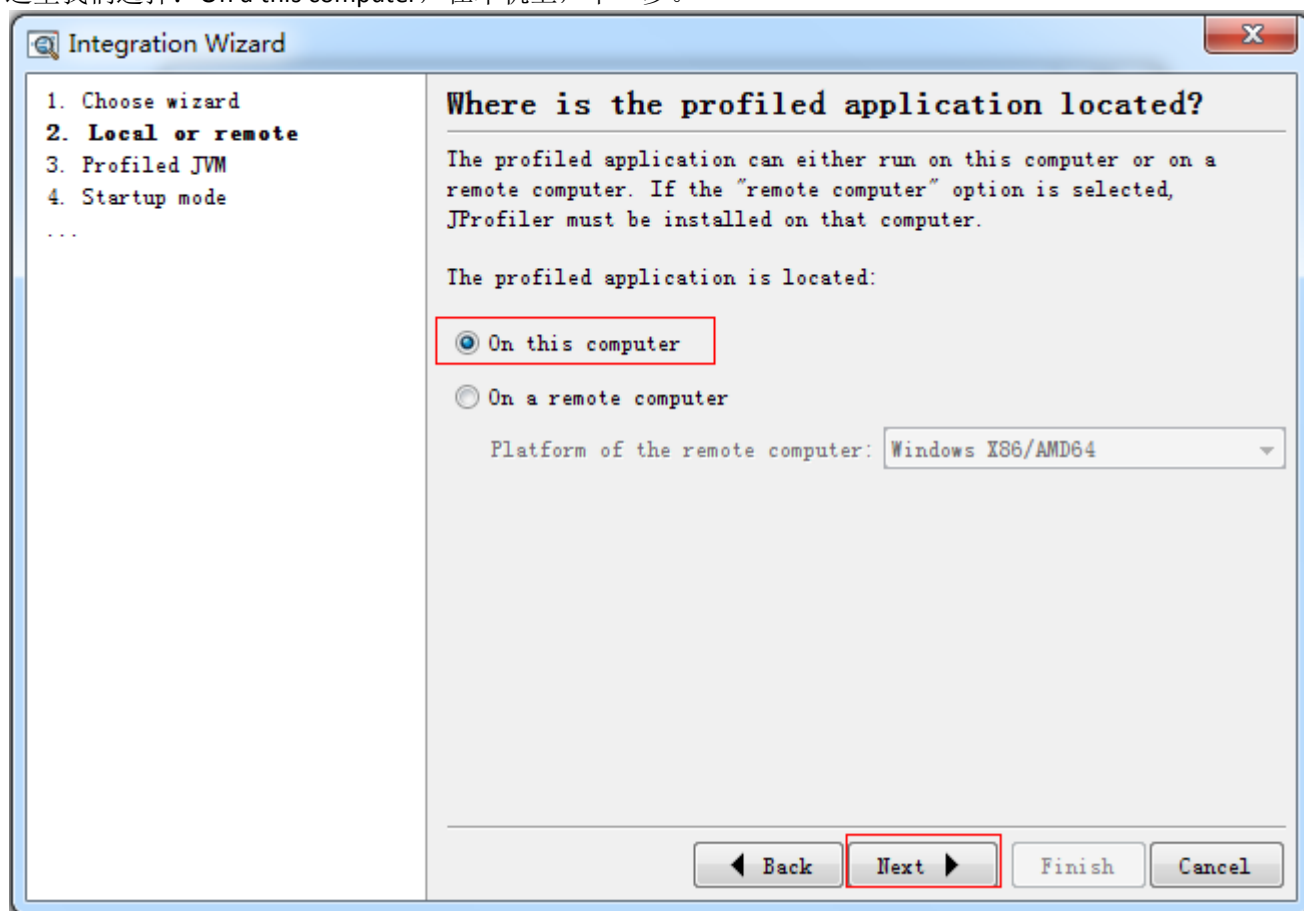


## 6.3 选择 Tomcat 容器的位置

选择是 Tomcat 在本机，还是在远程主机上。

当你想配置监视远程的 Tomcat 时，就可以选择 On a remote computer。

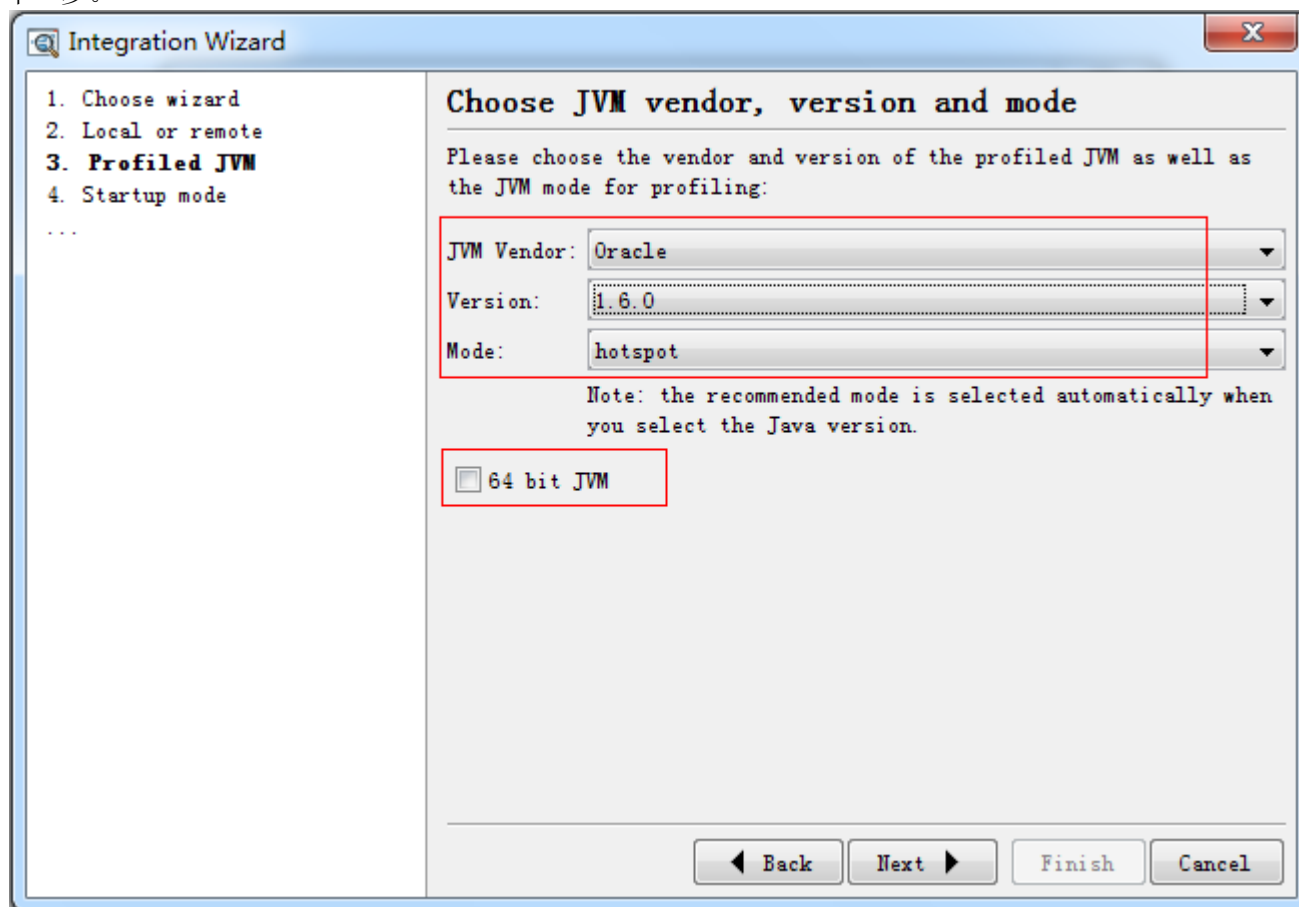
这里我们选择：On a this computer，在本机上，下一步。



## 6.4 选择虚拟机类型

因为所有的监视 JVM 的信息都是由 JVM 所提供的接口(JVM TI)给出的，这里要注意选择正确 JVM，我这里使用 Oracle(sun) 的 1.6.0 hotspot，你的 JVM 是 32 位不用打勾，是 64 位就要打勾。

下一步。

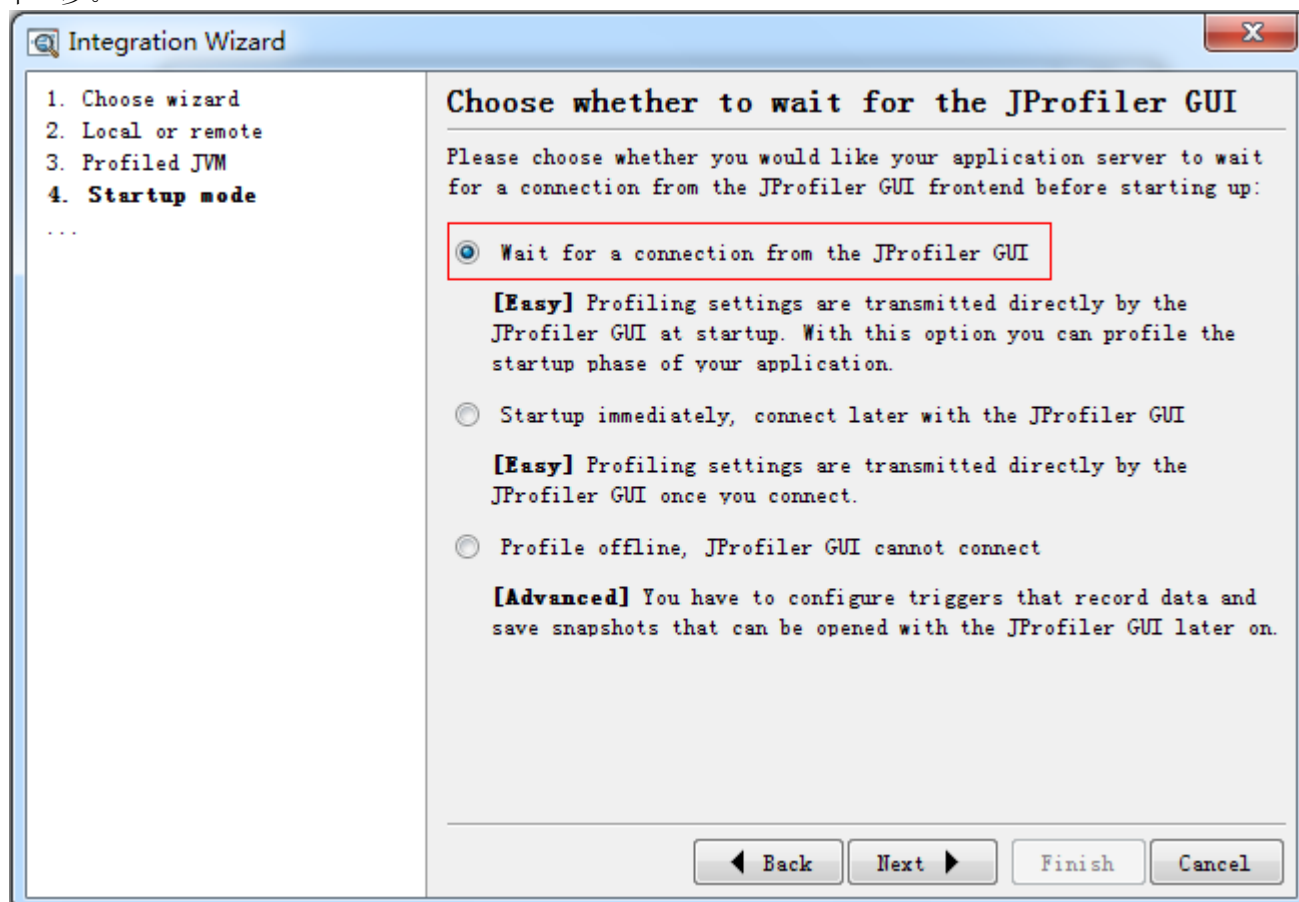


## 6.5 确认启动监控的方式

选第一项：等待模式

wait for a connection from the JProfiler GUI,此模式为，在启动容器（Weblogic/Tomcat）时，需要等 JProfiler 连接后才能启动；

下一步。

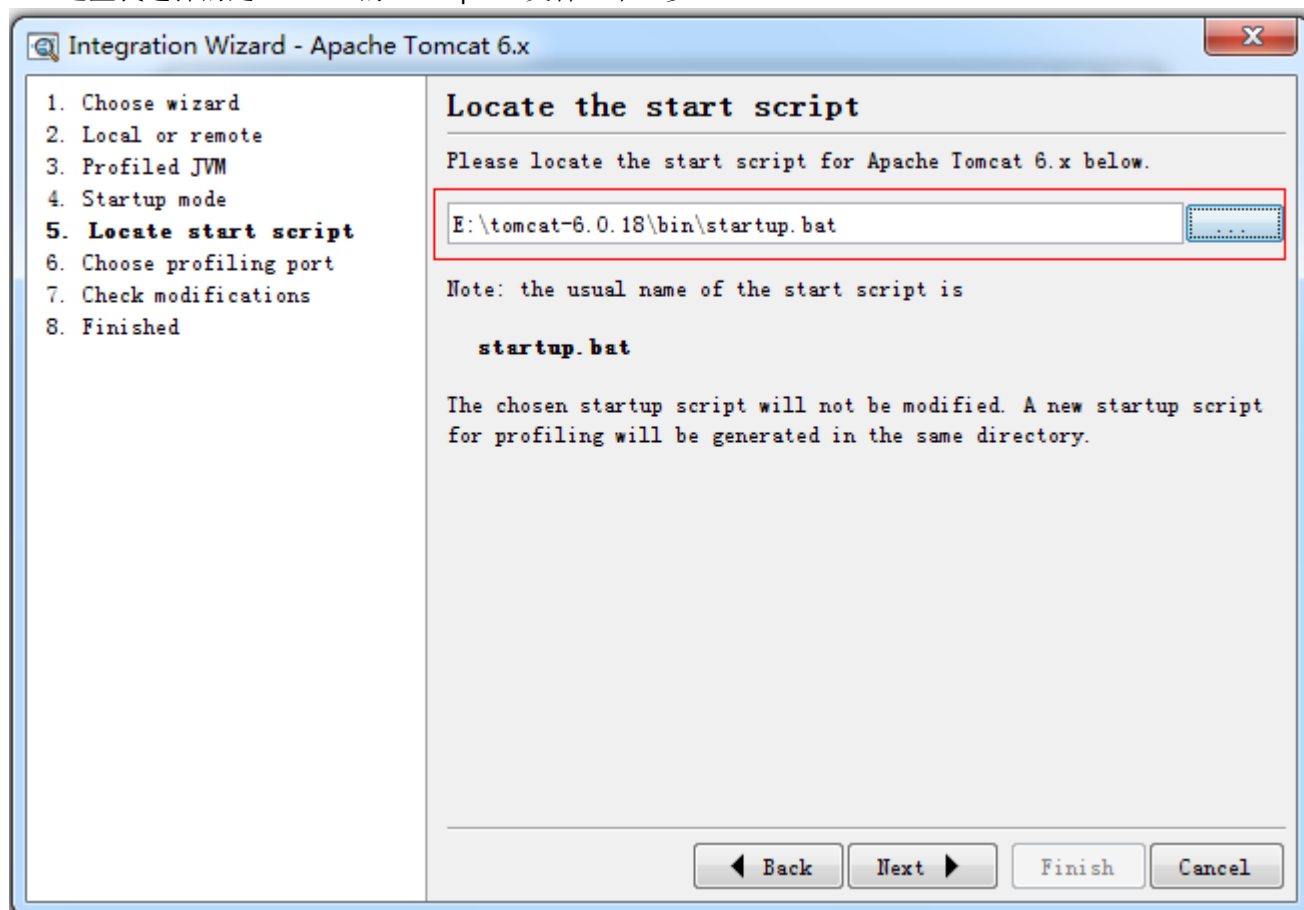


## 6.6 选择容器启动脚本

Tomcat 容器的启动可以由 jprofiler 来控制（可以手动控制），jprofiler 要修改启动文件，加入 JVM TI 参数实现监视目的。

选择好 Tomcat 启动脚本后 jprofiler 会自动为我们生成新的启动脚本在 Tomcat 的 bin 目录，文件名是 startup\_jprofiler.bat，通过这个文件来启动 Tomcat。

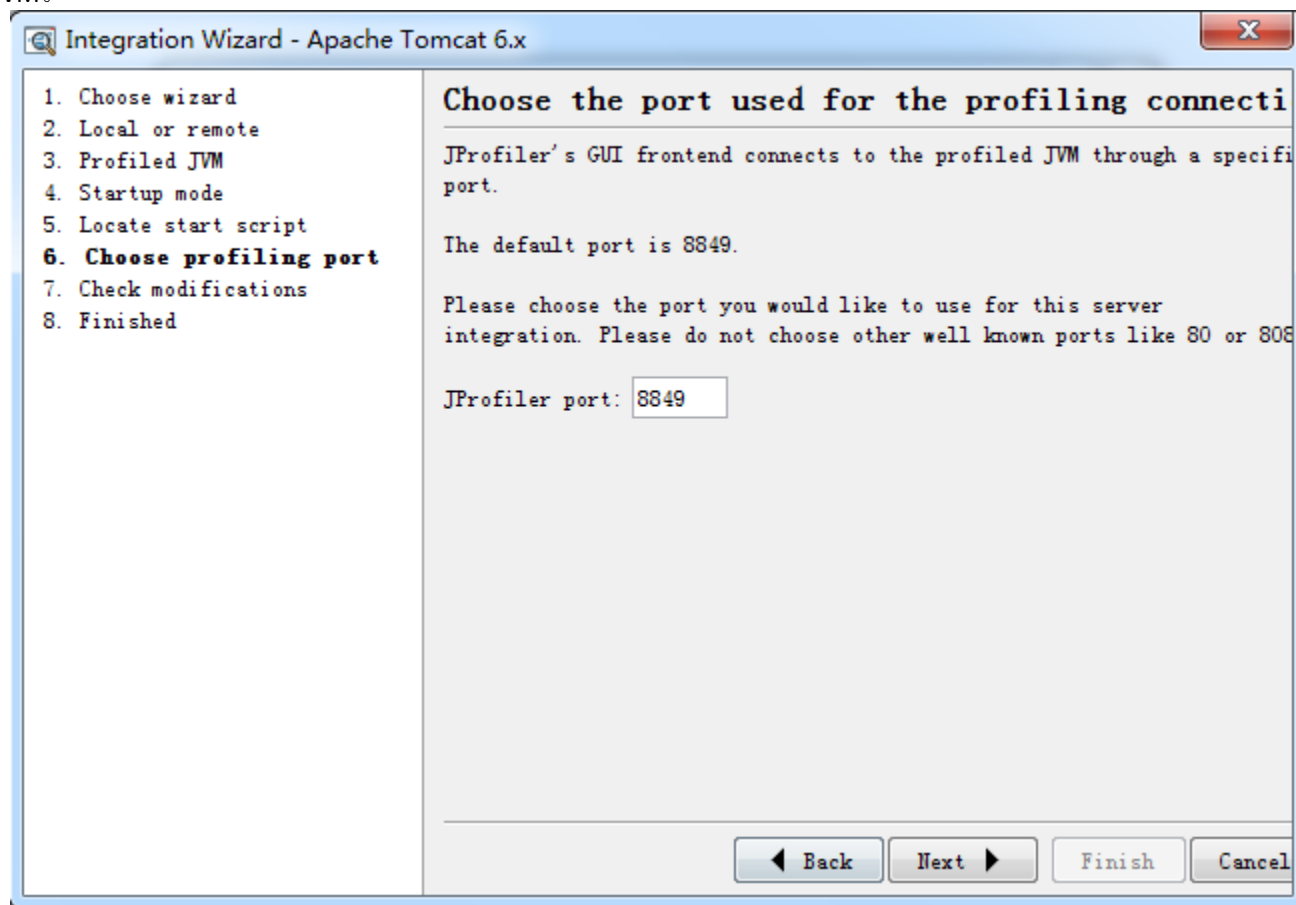
这里我选择的是 tomcat 的 startup.bat 文件。下一步。



## 6.7 选择监控端口

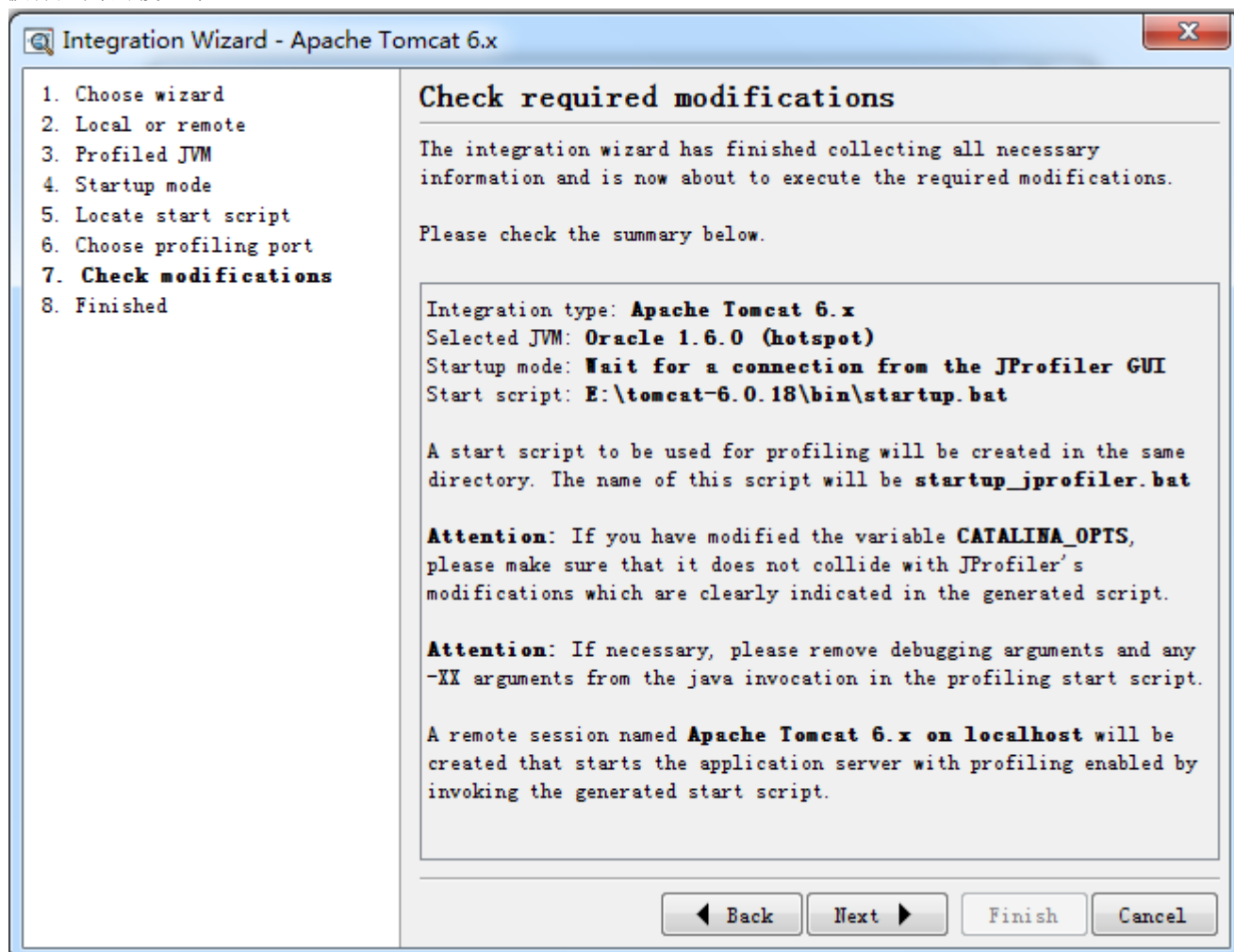
只要不冲突就行，这里我就使用默认的。

JProfiler GUI 在一个单独的 JVM 里启动，他与被监视的目标 jvm 之间的通讯是通过 socket 实现的，为了不干扰目标 JVM。



## 6.8 最后确认信息

信息的内容说：创建了一个启动脚本，在相同的目录里。可以用它的来启动。其中添加了 CATALINA\_OPTS 参数，别被你人为的覆盖了。



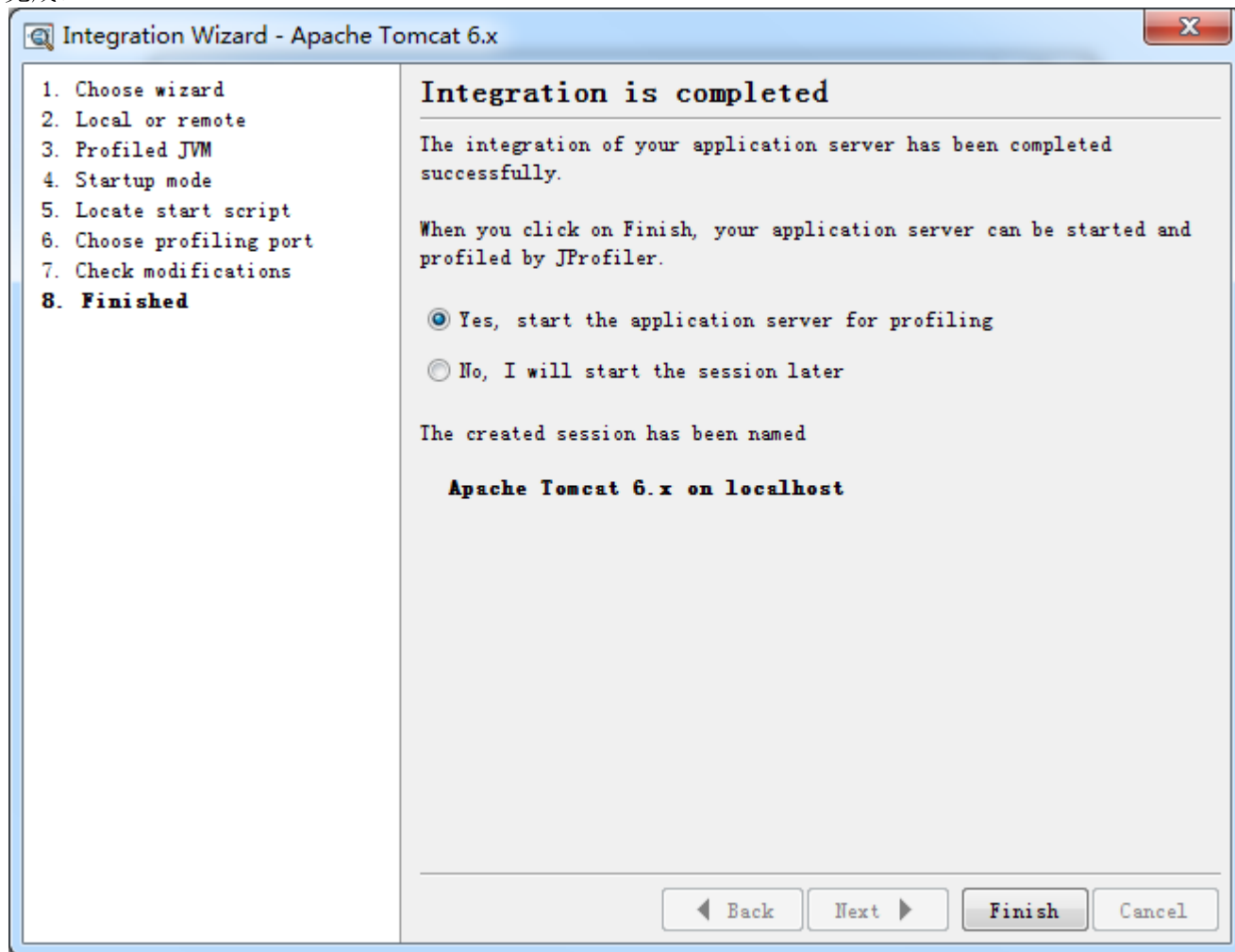
## 6.9 启动容器

这里选择立刻开始启动容器并进行监控，还是以后再启动。

立刻开始：启动窗口，并监视 JVM

以后再启动：在 Start Center->Open Session 中查找创建的 session 并启动。

手动启动：在 Tomcat 的 bin 目录下，新生成的 startup\_jprofiler.bat 文件，可以用来启动 Tomcat，其中只是加入了 JVM TI 的参数。执行 startup\_jprofiler.bat 来启动，如果 Jprofiler 不连接，就无法完成启动，用 Jprofiler 连接上，启动完成。



最后一步：

点击完成后，又出现一个窗口，是运行监控时的几个选项，不用动，使用默认就行，点击 ok 就开始运行了。

## 七、创建监视远程 Tomcat 的工程（等待模式）

### 基本思路：

- 1、服务端（远端）操作系统也要安装 JProfiler，因为服务端 Tomcat 启动时，JVM 加载 JProfiler 的模块。
- 2、JProfiler 在本地 Tomcat 的 bin 目录下（客户端要有一个与服务端版本号相同的 Tomcat），新生成的 startup\_jprofiler.sh 文件，其中添加了上面提到的 JVM 参数。把这个文件 copy 到远端的 Tomcat 服务器上，可以用来远端的启动 Tomcat。
- 3、客户端也可以没有 Tomcat，这样 JProfiler 就不能生成 startup\_jprofiler.sh 文件，而只是在向导的最后一步给出 JVM 的启动参数，你可以手动把这个参数添加到远程 Tomcat 的 startup.sh 或 catalina.sh 文件中，也可以达到目的。只是有点麻烦。

### 7.1 测试环境

服务器：RedHat Linux ， Tomcat6， Sun JDK 1.6.0\_29， JProfiler 7.1.2 for linux（安装包：jprofiler\_linux\_7\_1\_2.sh）

客户端：Windows 7， JProfiler 7.1.2 for windows（安装包：jprofiler\_windows\_7\_1\_2.exe）

### 7.2 JProfiler 软件下载地址

<http://www.ej-technologies.com/>

<http://www.ej-technologies.com/download/jprofiler/files.html>

（不需要注册就可以下载）

### 7.3 客户端 JProfiler 安装

在 Windows 上安装软件很容易。略。

### 7.4 服务器端 JProfiler 安装

把 jprofiler\_linux\_7\_1\_2.sh 上传到服务器，假设路径为 /opt/jprofiler7

执行以下命令：

```
# cd /opt/jprofiler7
# chmod +x *.sh
# ./jprofiler_linux_7_1_2.sh -c
```

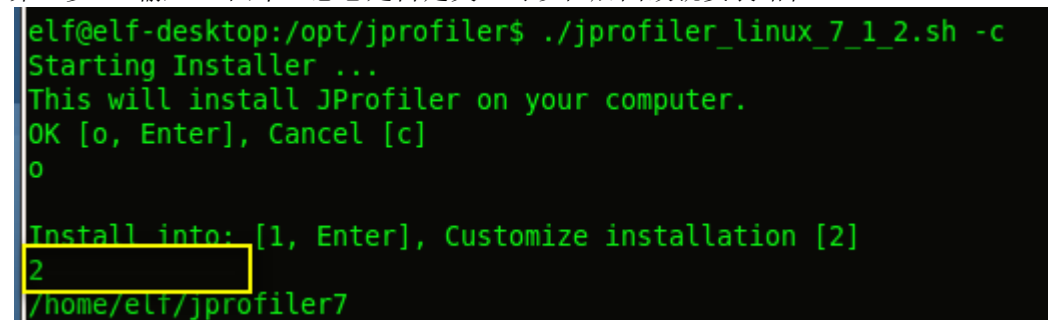
按照提示来安装，提示都很简单，不在多说。

注意，这里的 -c 意思是用字符方式来安装，如果机器上没有 图形界面 则加上该参数。

注意安装的路径的修改，看下图：

第一步： 输入 o 回车

第二步： 输入 2 回车，意思是自定义，可以在后面改就安装路径



```
elf@elf-desktop:/opt/jprofiler$ ./jprofiler_linux_7_1_2.sh -c
Starting Installer ...
This will install JProfiler on your computer.
OK [o, Enter], Cancel [c]
o
Install into: [1, Enter], Customize installation [2]
2
/home/elf/jprofiler7
```

安装路径选择 /opt/jprofiler7



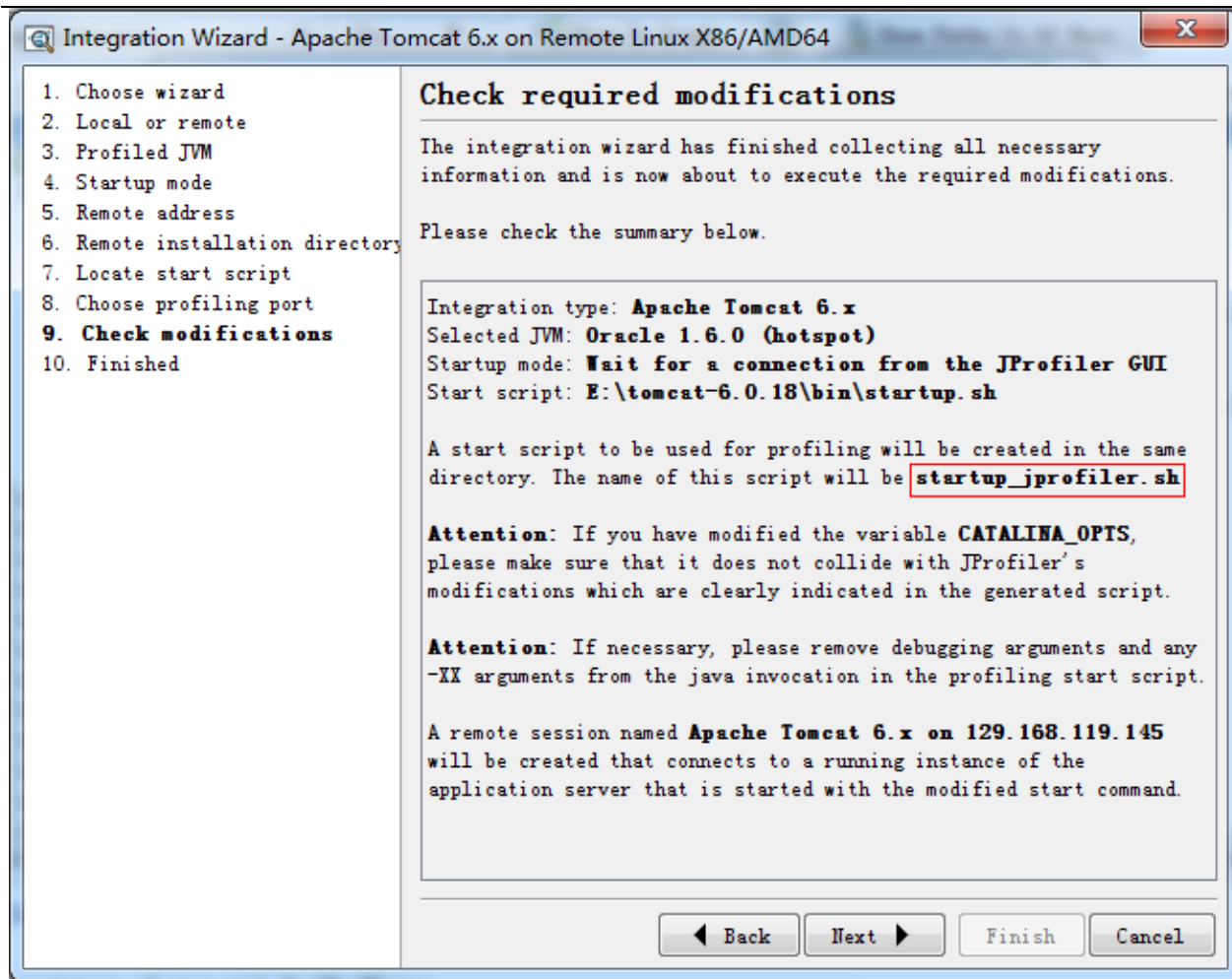
```
I accept the agreement
Yes [1], No [2]
1
Where should JProfiler be installed?
[/home/elf/jprofiler7]
/opt/jprofiler7
```

基本一路 OK 啊、同意啊 什么的。没有要求输入序列号。

## 7.5 客户端连接配置

前提：客户端有一个与服务端版本号相同的 Tomcat。

- 1). 运行 JProfiler 。忽略快速启动菜单。
  - 2). 选择 Session->Integration Wizard->New Server Integratation
  - 3). 选择 Apache Tomcat 6.x （不建议选择 with tomcat6.exe)
  - 4). 选择 On a remote computer; Platform of remote computer 选择 Linux x86/AMD 64;      Next
  - 5). 选择服务器的 JDK 环境，这里是：Oracle，1.6.0，hotspot，(注意是 32 位或 64 位)      Next
  - 6). 选择启动模式：这里选第一种 wait for a connection from the jprofiler GUI，等待模式      Next
  - 7). 输入服务器 IP ，      Next
  - 8). 输入服务器上的 jprofiler 的安装路径，如 /opt/jprofiler7 ，用于加载模块 。      Next
  - 9). 选择客户端本地 tomcat 的启动脚本，例如：E:\tomcat-6.0.18\bin\startup.sh，做为修改的样本。      Next
  - 10). 输入端口：这里是默认值 8849; Next
- 9). 这里会列出需要在服务器端做的配置:核心内容是说在 E:\tomcat-6.0.18\bin\目录生成了 startup\_jprofiler.sh，把它的 copy 到服务端 Tomcat 的相同目录中，用来启动 Tomcat。如下图：



## 八、创建监视远程 Tomcat 的工程（非等待模式）

请参照第七章做，在第 6 步时，选第二个选项。其它都一样。

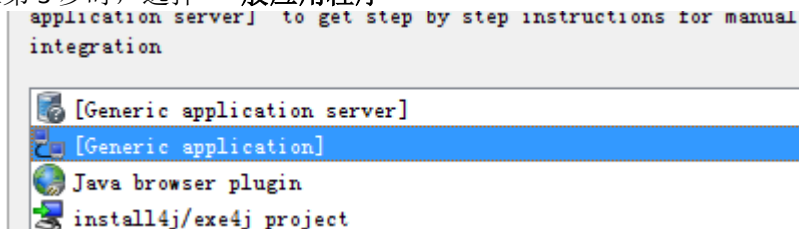
## 九、创建监视远程 Java 程序的工程

如果远程的服务端不是一个容器（Tomcat），而是由 main 方法启动的 Java 程序。

请参照第七章做，

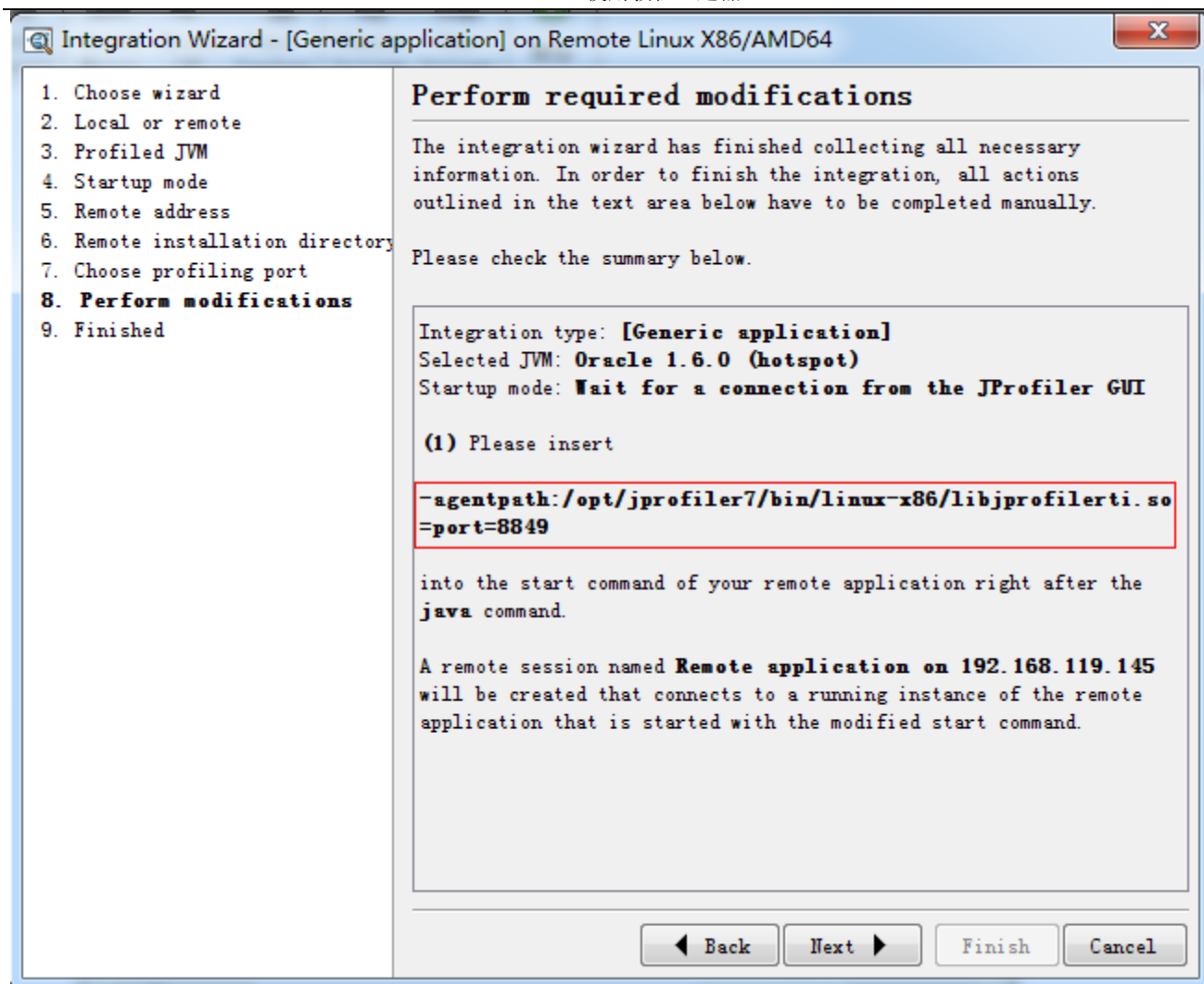
在第 2 步时，选择 Session->Integration Wizard->New Server Integration

在第 3 步时，选择 一般应用程序



其它基本差不多。

最后一步，结果，如下图：

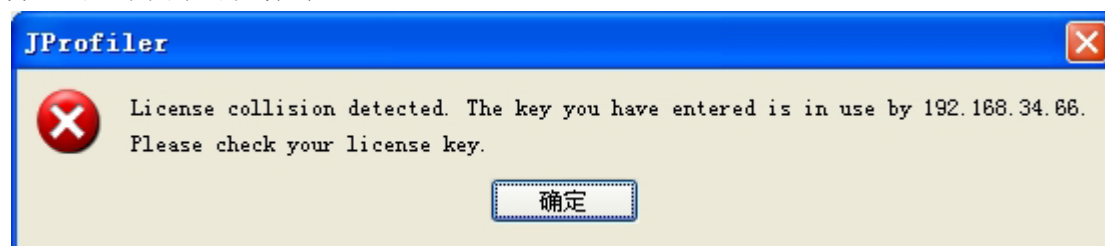


最后一步，生成 JVM 启动参数：`-agentpath:/opt/jprofiler7/bin/linux-x86/libjprofilerti.so=port=8849`  
在启动服务端 Java 程序时要加入这个参数。

## 十. 其它内容

### 10.0 局域内使用同一个 license key 问题

如果在局域内，有两个 JProfiler 使用同一个 license key，后打开的 JProfiler 会导致先前打开的 JProfiler 弹出以下窗口并退出。下图中可以看到 IP。



## 10.1 JProfiler 对程序性能的影响

开启 JProfiler 监视，对应用程序的性能会有多大的影响？通过的手上现在程序的对比发现 JProfiler 对被监视的应用程序的性能影响很小。可以通过下面两张图片看出来。

目标是一个网络通信程序，做两点间通信，分别列出未使用与使用 JProfiler 监视情况下运行结果。

未使用 JProfiler，程序原本的的性能结果

```
-----↓
并发数:1,每并发请求次数:10000,数据大小:4096字节↓
开始正式测试↓
总    用    时(s):6.178613000↓
每一次用时(ms):0.617831000↓
吞    吐    量(M/s):6.322211797↓
吞    吐    量(TPS):1618.486220127↓
-----↓

并发数:5,每并发请求次数:10000,数据大小:4096字节↓
开始正式测试↓
总    用    时(s):8.292631000↓
每一次用时(ms):0.827317000↓
吞    吐    量(M/s):23.552537186↓
吞    吐    量(TPS):6029.449519700↓
-----↓

并发数:10,每并发请求次数:10000,数据大小:4096字节↓
开始正式测试↓
总    用    时(s):16.344305000↓
每一次用时(ms):1.629833000↓
吞    吐    量(M/s):23.899762027↓
吞    吐    量(TPS):6118.339078964←
```

使用 JProfiler 后，程序的的性能结果

```
1 -----↓
2 并发数:1,每并发请求次数:10000,数据大小\:\:4096字节↓
3 开始正式测试↓
4 总    用    时(s):6.445190000↓
5 每一次用时(ms):0.644473000↓
6 吞    吐    量(M/s):6.060721251↓
7 吞    吐    量(TPS):1551.544640267↓
8 -----↓
9 并发数:5,每并发请求次数:10000,数据大小\:\:4096字节↓
10 开始正式测试↓
11 总    用    时(s):9.226549000↓
12 每一次用时(ms):0.921784000↓
13 吞    吐    量(M/s):21.168532243↓
14 吞    吐    量(TPS):5419.144254260↓
15 -----↓
16 并发数:10,每并发请求次数:10000,数据大小\:\:4096字节↓
17 开始正式测试↓
18 总    用    时(s):18.480306000↓
19 每一次用时(ms):1.845558000↓
20 吞    吐    量(M/s):21.137366448↓
21 吞    吐    量(TPS):5411.165810783↓
22 <
```