肥侠(邱小侠)
xiaoxia.qxx@alibaba-inc.com
阿里巴巴，客户体验驱动创新中心，架构师

JAVA DEBUG那点事

Agenda

- Java Debugging
- JPDA
  - JVMTI
  - JDWP
  - JDI
- 例子

调试的定义

- 调试（De-bug），又称除错，是发现和减少计算机程序或电子仪器设备中程序错误的一个过程。

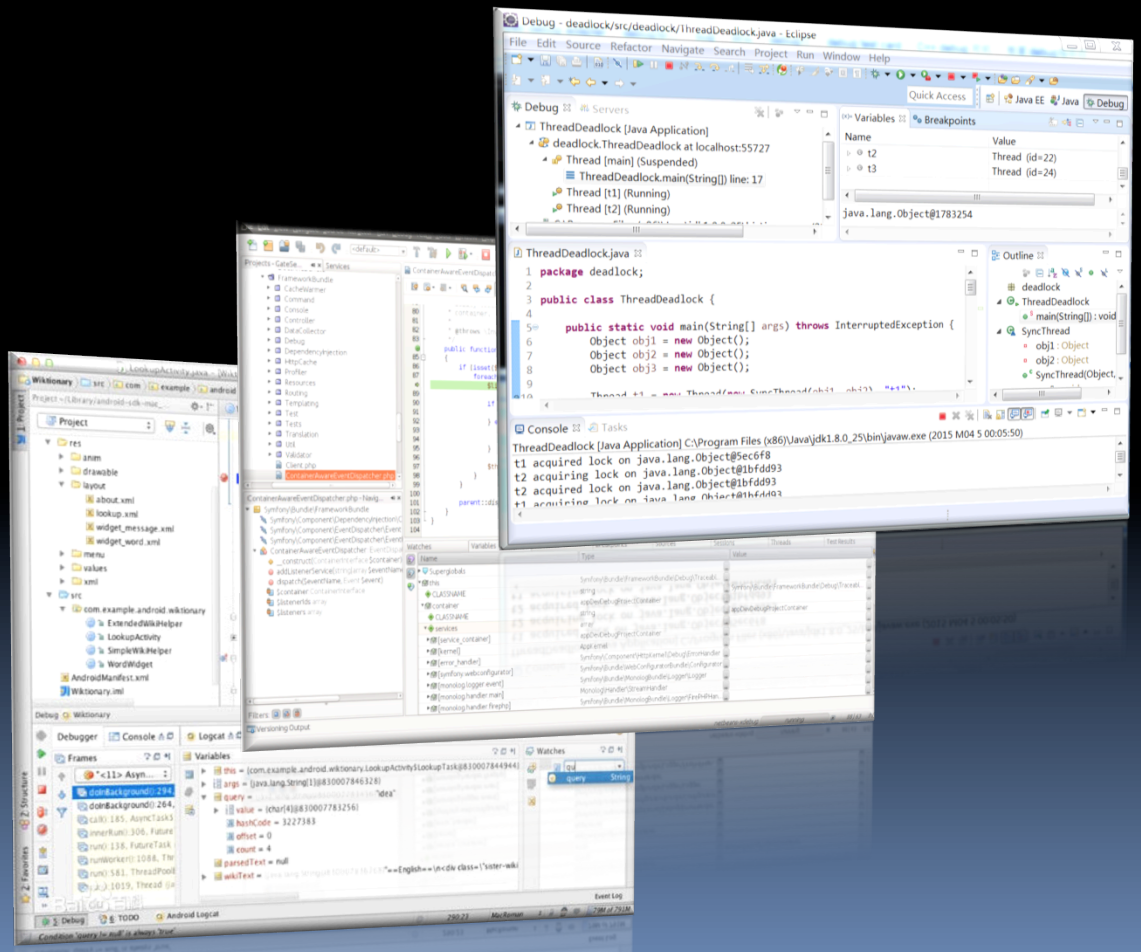
- 调试的基本步骤
  - 发现程序错误的存在
  - 以隔离、消除的方式对错误进行定位
  - 确定错误产生的原因
  - 提出纠正错误的解决办法
  - 对程序错误予以改正，重新测试

http://zh.wikipedia.org/wiki/调试

Java程序员如何Debug？

IDE

- Eclispe
- Netbeans
- IntelliJ



JDK Tools

Tool Name	Brief Description
jdb	The Java debugger
jvisualvm	A graphical tool that provides detailed information about the Java technology-based applications (Java applications) while they are running in a Java Virtual Machine. Java VisualVM provides memory and CPU profiling, heap dump analysis, memory leak detection, access to MBeans, and garbage collection.
jconsole	A JMX-compliant graphical tool for monitoring a Java virtual machine. It can monitor both local and remote JVMs. It can also monitor and manage an application.
jps	JVM Process Status Tool - Lists instrumented HotSpot Java virtual machines on a target system.
jstat	JVM Statistics Monitoring Tool - Attaches to an instrumented HotSpot Java virtual machine and collects and logs performance statistics as specified by the command line options.
jinfo	Configuration Info for Java - Prints configuration information for a given process or core file or a remote debug server.
jhat	Heap Dump Browser - Starts a web server on a heap dump file (eg, produced by jmap -dump), allowing the heap to be browsed.
jmap	Memory Map for Java - Prints shared object memory maps or heap memory details of a given process or core file or a remote debug server.
jstack	Stack Trace for Java - Prints a stack trace of threads for a given process or core file or remote debug server.

还有许多第三方debug/profile工具.....

Reference： http://docs.oracle.com/javase/6/docs/technotes/tools/index.html

常用线上问题诊断流程

1 可以通过 top 和 vmstat 查看load状况

2 通过ps -eLf | grep java | wc –l 统计java线程

3 通过jstack查看线程都在干什么

4 通过jstat 查看java gc执行状况

5 通过jmap 查看堆内存

6 通过查看日志判断系统慢在什么地方

7 通过查看日志判断cache ，数据库或者依赖的其他系统

8 利用btrace定位怀疑点

Debug的核心

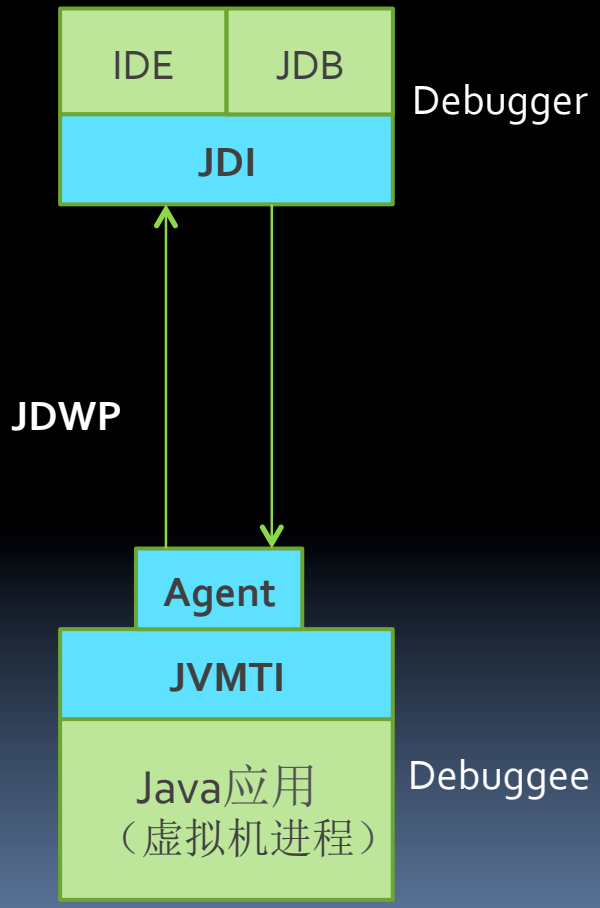
- 调试的基本步骤
  - 发现程序错误的存在
  - 以隔离、消除的方式对错误进行定位—— 通过Trace虚拟机行为
  - 确定错误产生的原因
  - 提出纠正错误的解决办法
  - 对程序错误予以改正，重新测试

那么问题来了

JPDA - Java的调试体系

- JVM TI - Java VM Tool Interface
  - 虚拟机对外暴露的接口，包括debug和profile
- JDWP - Java Debug Wire Protocol
  - 调试器和应用之间通信的协议
- JDI - Java Debug Interface
  - Java库接口，实现了JDWP协议的客户端，调试器可以用来和远程被调试应用通信

注：每个模块都可以自己实现(JVMTI外)



类比：

- *IDE+JDI = 浏览器*
- *JDWP = HTTP*
- *JVMTI = RESTful接口*
- *Debugee虚拟机= REST服务端*

JVMTI – JPDA基础

- 虚拟机信息
  - 堆上的对象
  - 线程和栈信息
  - 所有的类信息
  - 系统属性，运行状态
- 调试行为
  - 设置断点
  - 挂起现场
  - 调用方法
- 事件通知
  - 断点发生
  - 异步调用

http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/jvmti.html

JVMTI – JPDA基础

入口

初始化

堆操作

线程操作

事件回调

```
JNIEXPORT jint JNICALL Agent_OnLoad(JavaVM *vm, char *options, void *reserved)
```

```
jvmtiEnv *jvmti;
(*jvm)->GetEnv(jvm, &jvmti, JVMTI_VERSION_1_0);
```

```
err = (*jvmti)->GetCapabilities(jvmti, &capa); // 取得 jvmtiCapabilities 指针。
 if (err == JVMTI_ERROR_NONE) {
           if (capa.can_redefine_any_class) { ... }
           } // 查看是否支持重定义类
```

```
jvmtiError IterateThroughHeap(jvmtiEnv* env,
    jint heap_filter,
    jclass klass,
    const jvmtiHeapCallbacks* callbacks,
    const void* user_data)// 遍历整个 heap
```

```
jvmtiError GetOwnedMonitorInfo(jvmtiEnv* env,
    jthread thread,
    jint* owned_monitor_count_ptr,
    jobject** owned_monitors_ptr)
```

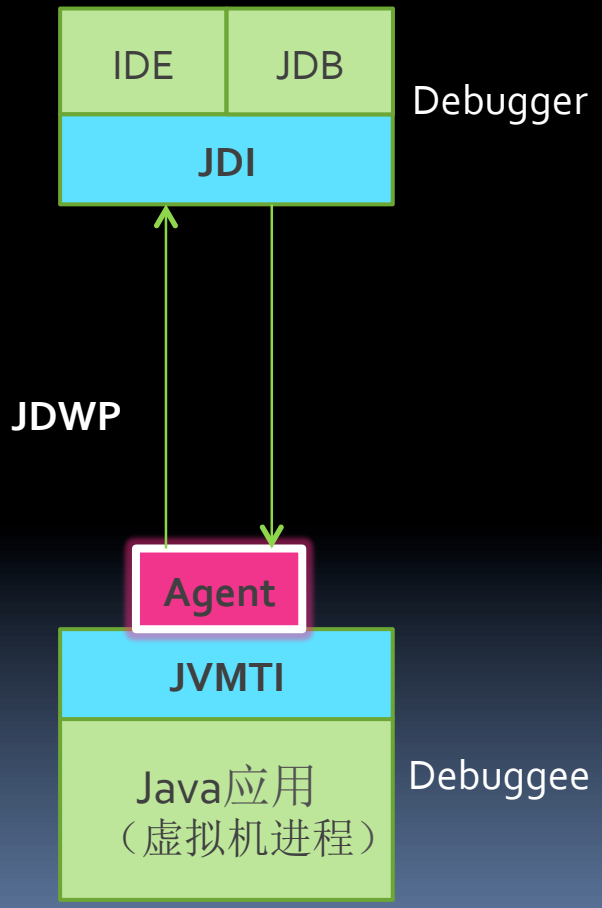
```
jvmti->SetEventCallbacks(eventCallBacks, sizeof(eventCallBacks));
```

```
jvmtiEventCallbacks eventCallBacks;
memset(&ecbs, 0, sizeof(ecbs)); // 初始化
eventCallBacks.ThreadStart = &HandleThreadStart; // 设置函数指针
...
```

```
jvmtiError SetBreakpoint(jvmtiEnv* env,
    jmethodID method,
    jlocation location)
```

JVMTI Agent - Java调试的核心

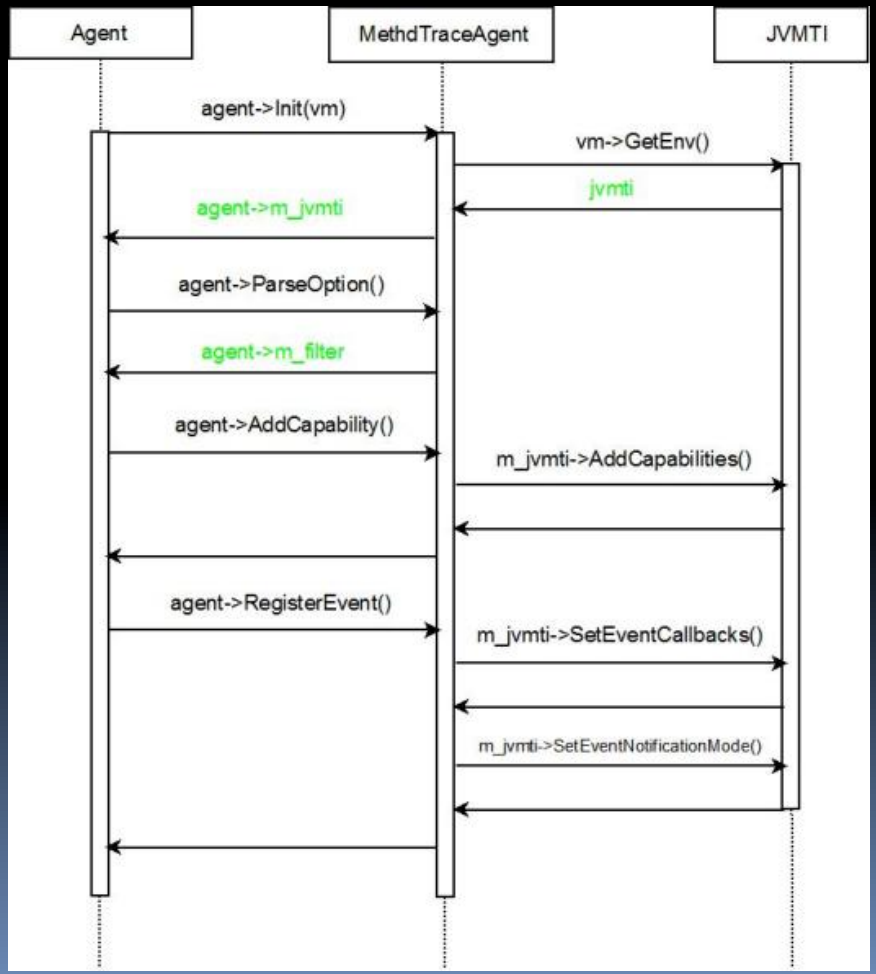
- C/C++实现
- 被虚拟机以动态库的方式加载
- 调用本地JVMTI提供调试能力
- 实现JDWP和JDI通信（socket/shmem）



JVMTI Agent例子

```
JNIEXPORT jint JNICALL Agent_OnLoad(JavaVM *vm, char *options, void *res
{
    ...
    MethodTraceAgent* agent = new MethodTraceAgent();
    agent->Init(vm);
    agent->ParseOptions(options);
    agent->AddCapability();
    agent->RegisterEvent();
    ...
}
class MethodTraceAgent
{
    public:
        void Init(JavaVM *vm) const throw(AgentException);
        void ParseOptions(const char* str) const throw(AgentException);
        void AddCapability() const throw(AgentException);
        void RegisterEvent() const throw(AgentException);
        ...

    private:
        ...
        static jvmtiEnv * m_jvmti;
        static char* m_filter;
};
```



例子

- JVMTI MethodTrace
（几乎可以控制虚拟机的任何行为）

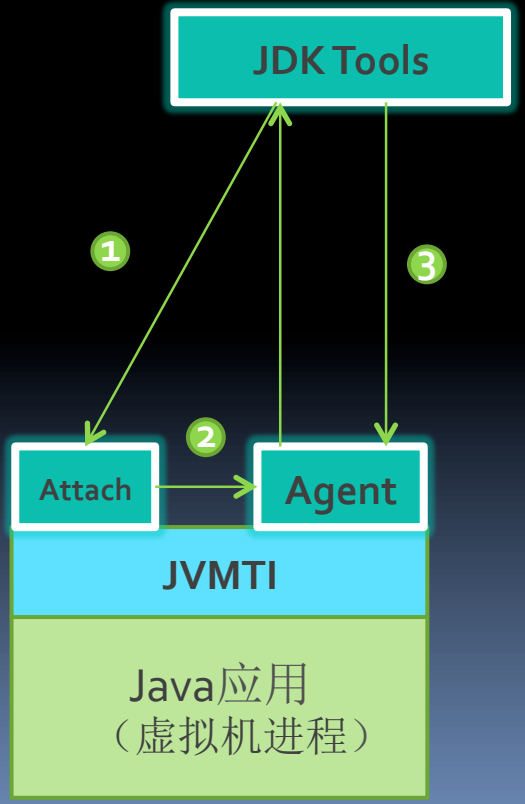
Agent加载

- 虚拟机启动初期加载
  - java -agentlib:<agent-lib-name>=<options> JavaClass
    - Linux从LD_LIBRARY_PATH找so 或 Windows从PATH找dll
  - java -agentpath:<path-to-agent>=<options> JavaClass
  - Agent_OnLoad <-> 入口函数
- 动态加载
  - Attach api
  - Signal Dispatcher和Attach Listener线程（jconsole）
  - Agent_OnAttach <-> 入口函数

Attach动态加载机制

- Jdktools（jconsole/jstack）是如何工作的？
- Instrument包如何工作的？
- Btrace如何工作？（profiling）
  - Attach -> load -> instrument

```
static AttachOperationFunctionInfo funcs[] = {
  { "agentProperties", get_agent_properties },
  { "datadump", data_dump },
  { "dumpheap", dump_heap },
  { "load", JvmtiExport::load_agent_library },
  { "properties", get_system_properties },
  { "threaddump", thread_dump },
  { "inspectheap", heap_inspection },
  { "setflag", set_flag },
  { "printflag", print_flag },
  { "jcmd", jcmd },
  { NULL, NULL }
};
```



例子

- 模拟jinfo命令

(可以用类似方法实现所有jdk工具)

JDWP Agent

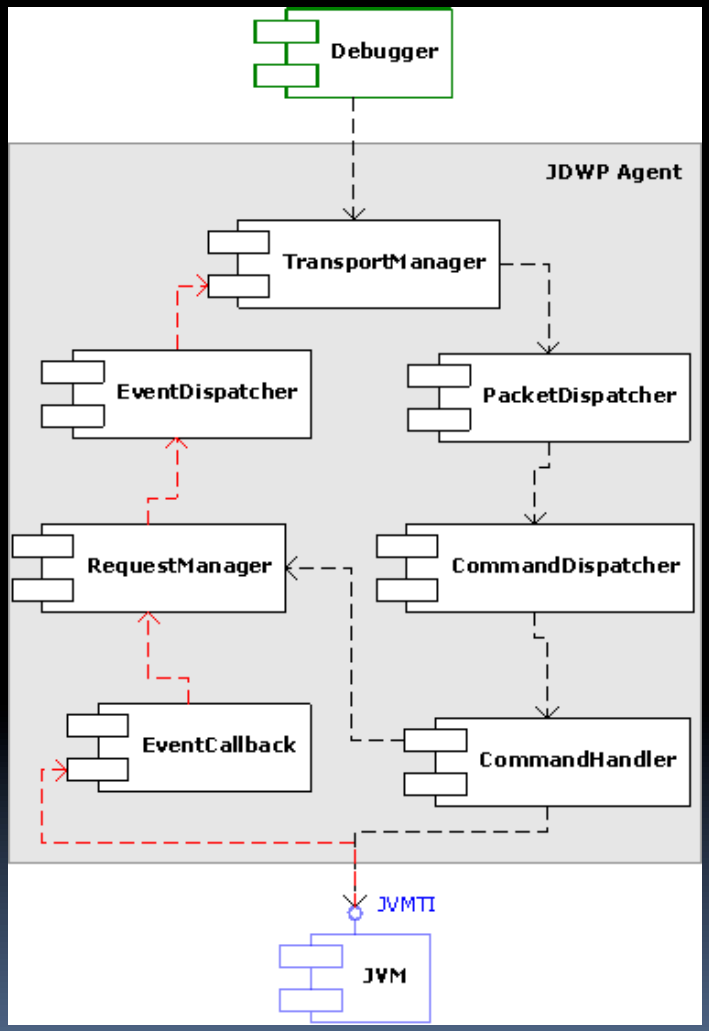
命令包格式

Length	Id	Flags	Command	
4	4	1	1	1
			Command Set	

回复包格式

Length	Id	Flags	Error Code
4	4	1	2

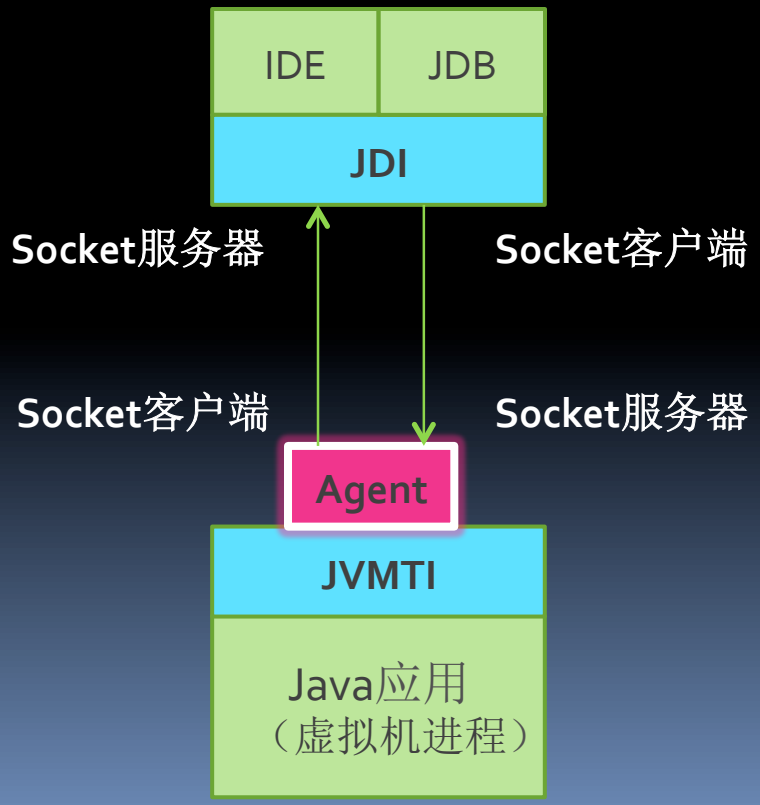
Flags 的值来判断接收到的 packet 是 command 还是 reply



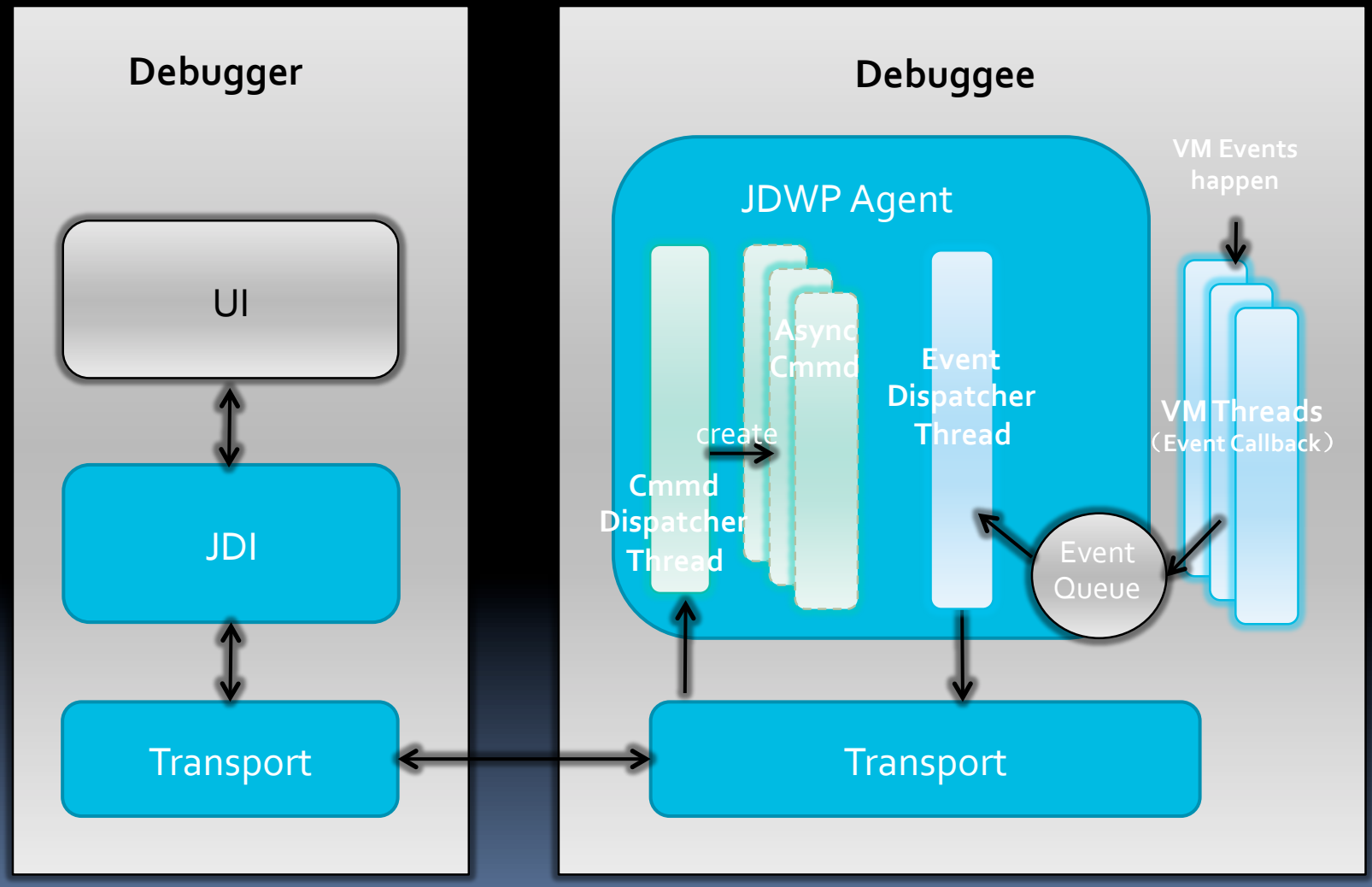
Reference: http://svn.apache.org/repos/asf/harmony/enhanced/java/trunk/jdktools/modules/jpda/doc/JDWP_agent.htm

JDWP的双向连接

- 参数
  - agentlib:jdwp=transport=dt_socket,server=y,address=8000
  - agentlib:jdwp=transport=dt_socket,address=myhost:8000

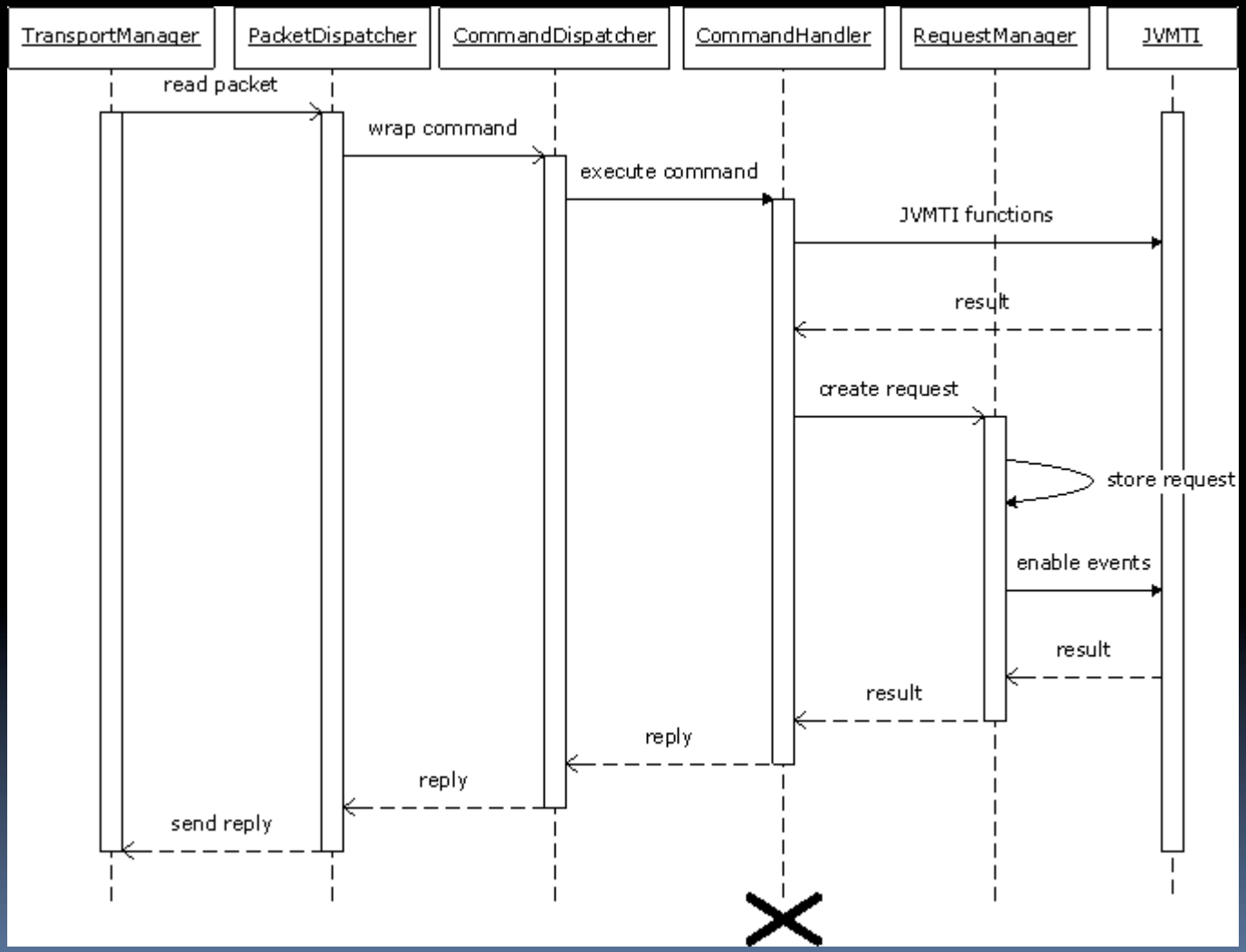


JDWP Agent



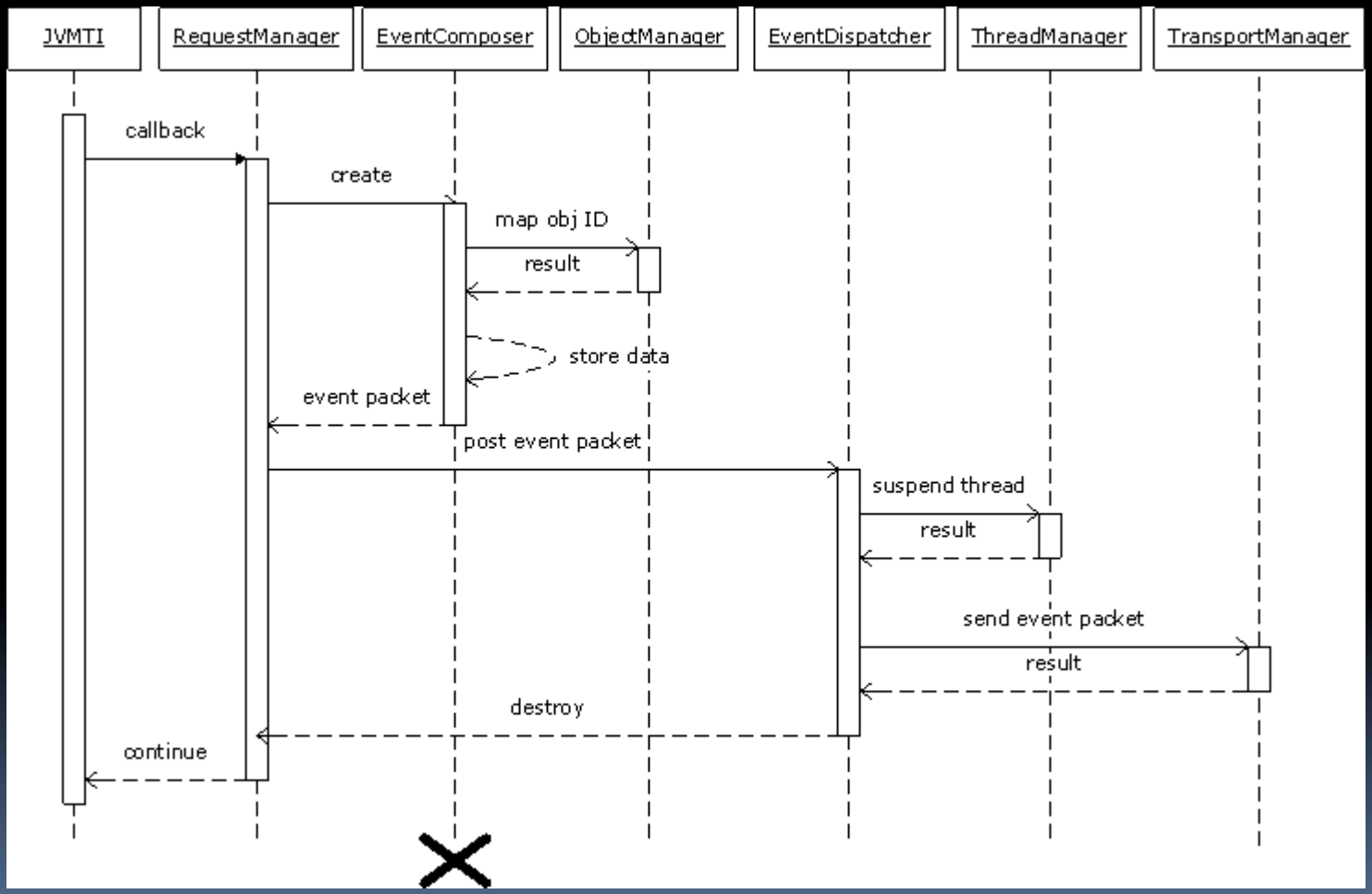
Reference: http://svn.apache.org/repos/asf/harmony/enhanced/java/trunk/jdktools/modules/jpda/doc/JDWP_agent.htm

JDWP – 简单的“命令-回复”



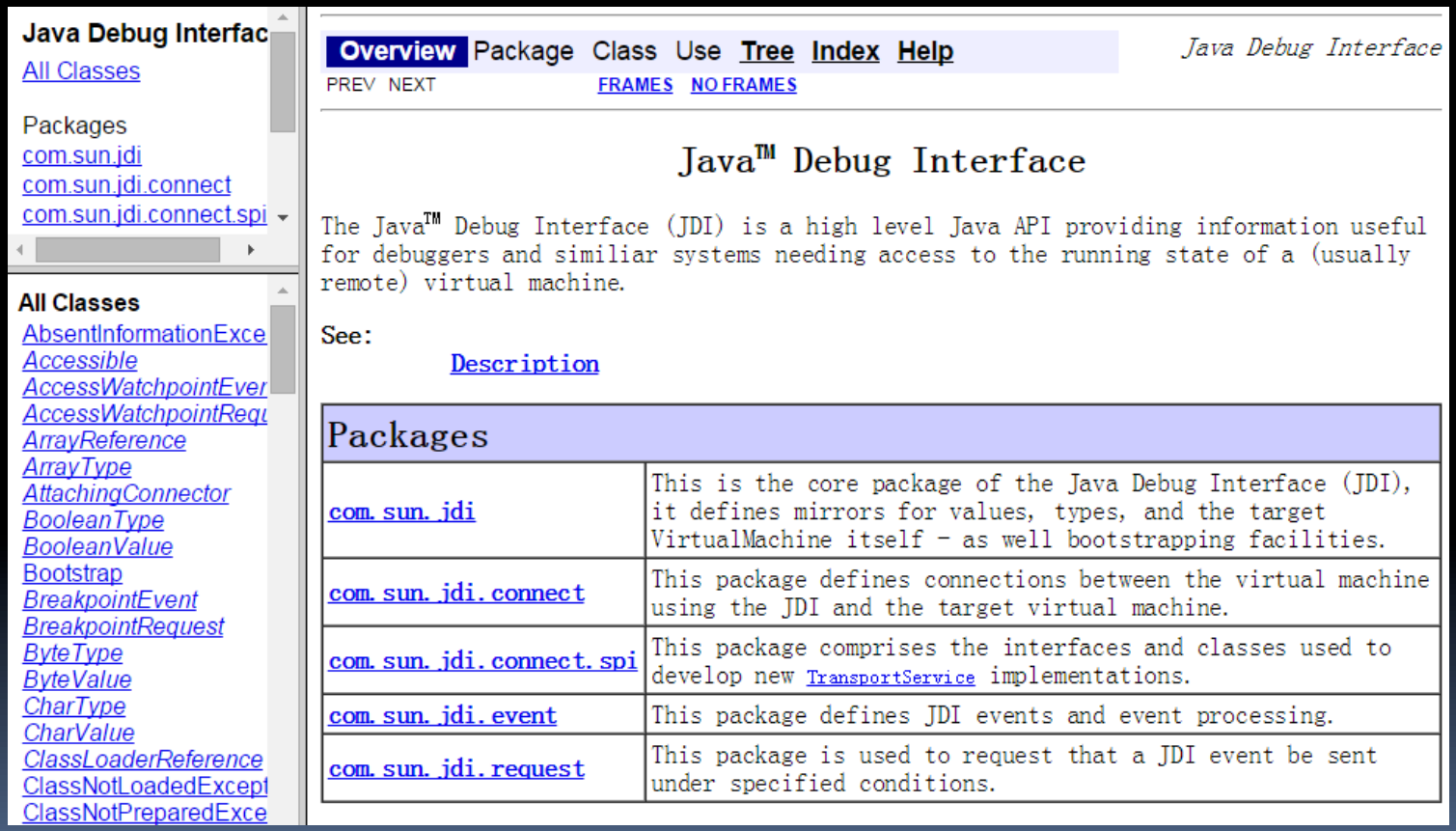
Reference： http://svn.apache.org/repos/asf/harmony/enhanced/java/trunk/jdktools/modules/jpda/doc/JDWP_agent.htm

JDWP – 事件回调



Reference： http://svn.apache.org/repos/asf/harmony/enhanced/java/trunk/jdktools/modules/jpda/doc/JDWP_agent.htm

JDI – java调试客户端接口



例子

- JDI MethodTrace
（可以自己写一个自动debugger）

Alibaba提供的诊断工具

- Tsar
  - https://github.com/alibaba/tsar
- HouseMD
  - https://github.com/CSUG/HouseMD/wiki/UserGuideCN
- Greys
  - https://github.com/oldmanpushcart/greys-anatomy

参考资料

- JPDA documentation
  - http://java.sun.com/products/jpda/index.jsp
- JDWP specification
  - http://java.sun.com/j2se/1.5.0/docs/guide/jpda/jdwp-spec.html
- JVMTI specification
  - http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/jvmti.html
- JDI specification
  - http://java.sun.com/j2se/1.5.0/docs/guide/jpda/jdi/index.html
- Developerworks
  - http://www.ibm.com/developerworks/cn/views/java/libraryview.jsp?search_by=深入+java+调试体系
- JVMTI Agent
  - http://jm-blog.aliapp.com/?p=756
- Attach API
  - http://my.oschina.net/xianggao/blog/364494
- Apache Harmony
  - http://svn.apache.org/repos/asf/harmony/enhanced/java/trunk/jdktools/modules/jpda/doc/JDWP_agent.htm

邮箱：xiaoxia.qxx@alibaba-inc.com