

Re: Determining the Main Class

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2008-08/msg01223.html>

- *From:* Daniele Futturovic <da.futt.news@xxxxxxxxxxxxxxxx>
 - *Date:* Mon, 11 Aug 2008 20:55:47 +0200
-

On 11/08/2008 19:54, Stefan Ram allegedly wrote:

Jason Cavett <jason.cavett@xxxxxxxx> writes:

Is there a way to determine (maybe through reflection or something?) where a Java application is being run from (what class contains the main method)?

Examine the stack trace.

```
public class Main { public static void method() { System.out.println
( java.util.Arrays.toString
( java.lang.Thread.currentThread().getStackTrace() )); }
public static void main( final java.lang.String[] args ) { Main.method(); }}
```

```
[java.lang.Thread.getStackTrace(Unknown Source), Main.method(Main.java:24),
Main.main(Main.java:28)]
```

In the stack trace above, it is the last entry. This might not always be so. Also, there might be other methods with the signature of main in the stack trace. So, some care has to be taken. Often, it should be the last method in the stack trace with the signature of main.

The code above also assumes that the current thread is the main thread.

I've investigated that a bit, trying to get the main Thread through searching the ThreadGroups, and also via Thread.getAllStackTraces.

There's a problem with that approach. To wit, that if the main Thread isn't **active**, you don't get it either way (at least that's what my tests indicate).

I would venture say that in a typical application, the main Thread isn't active. So this approach might not work.

Re: Determining the Main Class

As for searching for defined classes with a main method, that's pointless, since many classes can have a main defined.

As Mark said, it would work with a JAR (via its manifest). Ideally, you'd have to know the command the java executable was started with to deal with all cases...

--
DF.

.