Logging & Debugging von Oracle Mobile (MAF) Applikationen

Dr. Jürgen Menge

ORACLE Deutschland B.V. & Co. KG

Schlüsselworte

Oracle Mobile Application Framework, MAF, Logging, Debugging, Android, iOS, Java

Einleitung

Entwickelt und testet man Applikationen für mobile Endgeräte, können sehr unterschiedliche Fehlersituationen auftreten. Diese können ihre Ursachen in der Applikation selbst, aber auch in der Netzwerkverbindung zum Backend oder im Zusammenspiel von Backend und mobiler Applikation haben. Deshalb sind für den Entwickler leistungsfähige Hilfsmittel zum Debuggen und Protokollieren (Logging) notwendig.

Mit der aktuellen Version 2.0.1 des Oracle Mobile Application Framework (MAF) können mobile Apps für iOS (iPhone, iPad) und Android (Smartphones, Tablets) erstellt werden. Neben der Funktionalität des Oracle MAF Frameworks stellen beide Mobilplattformen eigene Werkzeuge zur Verfügung. Damit ergibt sich folgende Matrix:

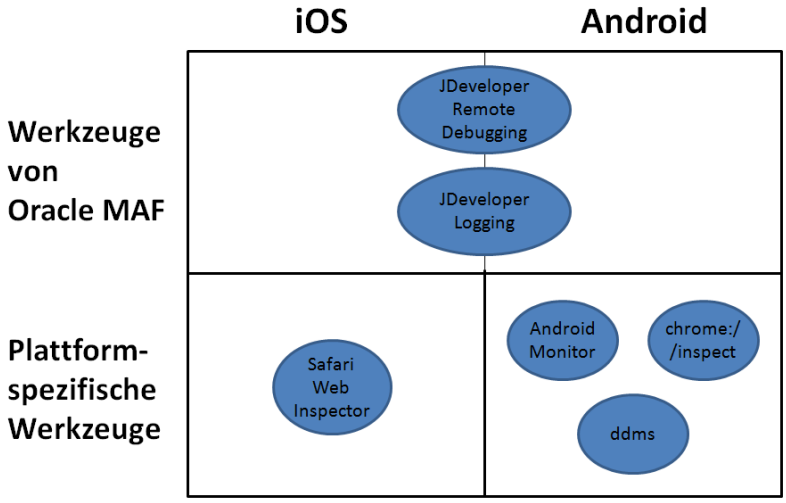

Abbildung 1: Werkzeuge zum Logging & Debugging von Oracle MAF Applikationen

Der folgende Vortrag wird sich auf den rechten Teil der Matrix (Android) konzentrieren und dabei vor allem die Möglichkeiten des Mobile Application Frameworks vorstellen.

Logging

Um ein Problem einzugrenzen, ist die Ausgabe von Nachrichten in das Standardausgabe-Protokoll sinnvoll. Dies ist bei Oracle MAF auch deshalb wichtig, da beim Remote Debugging (s. folgender Abschnitt) keine Breakpoints für deklarative Komponenten gesetzt werden können.

Für die Ausgabe der Meldungen gibt es zwei Möglichkeiten:

1. Nachrichten können mittels *system.out.println()* aus dem Java Code bzw. mitels *console.log()* aus dem JavaScript Code ausgegeben werden.

2. Alternativ kann ein Logger (*com.sun.util.logging.**) verwendet werden. Dieser kann über einen *ActionListener* aktiviert werden, der für eine deklarative Komponente (z.B. einen Button) definiert wird.

Das Logging muss in der *Datei <app>/src/META-INF/logging.properties* innerhalb der Applikation konfiguriert werden (Schweregrad, Logger).

Für die Auswertung der erzeugten Protokolldatei gibt es mehrere Möglichkeiten, die sich auch darin unterscheiden, ob der Test auf einem nativen Gerät oder im Simulator/Emulator durchgeführt wurde.

1. Bei Android wird die Logdatei im Root-Verzeichnis der internen Speicherkarte (*/mnt/sdcard*) mit dem Namen <*application_name.txt*> abgelegt. Eine evtl. bereits vorhandene Datei gleichen Namens wird als Backup (.bak) aufgehoben.

2. Alternativ kann im Fall von Android der Monitor des Android SDK oder das Utility *ddms* verwenden werden, um die Nachrichten anzuzeigen. Bei diesen Tools besteht die Möglichkeit, aus der Vielzahl von Meldungen diejenigen herauszufiltern, an denen man speziell interessiert ist.

Debugging

Für Entwickler sind die Möglichkeiten des Debuggens von Applikationen von besonderem Interesse. So erlauben der Oracle JDeveloper und das Oracle Enterprise Pack for Eclipse (OEPE) ein Remote Debugging von mobilen Applikationen. Möglich ist das Debuggen von Java-Code und JavaScript/CSS. Nicht unterstützt werden Breakpoints in deklarativen Komponenten und die Evaluierung von Ausdrücken der Expression Language (EL).
Das Debugging wird zunächst in der Datei *<app>/src/META-INF/cvm.properties* innerhalb der Applikation konfiguriert:

```
java.debug.enabled=true
java.debug.port=<port>
```

Anschließend muss die Applikation im Debug Modus auf dem Gerät oder im Simulator/Emulator deployed werden.

Vor dem Aufruf des Debuggers muss dieser einmalig konfiguriert werden (siehe Abbildung 2.)

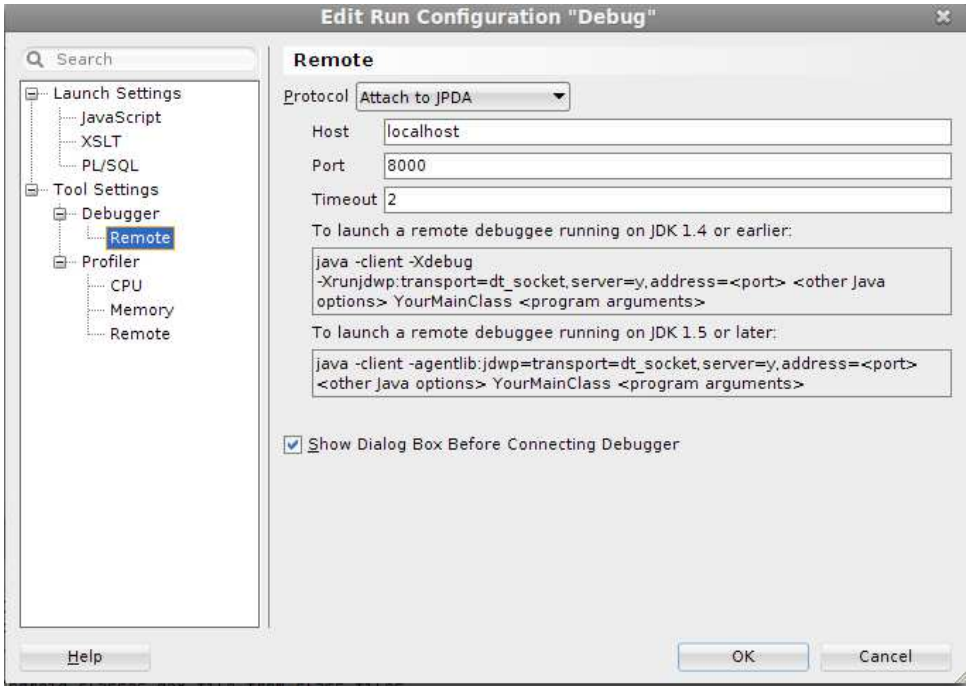

Abbildung 2: Konfiguration des Remote Debugging

Wird die Applikation auf dem Gerät oder Simulator/Emulator aufgerufen, wartet sie auf den Start des Debuggers. Zunächst muss jedoch ein Port Forwarding angestoßen werden. Bei Android ist das z.B. eines der beiden folgenden Kommandos:

```
adb -e forward tcp:<port> tcp:<port>   Emulator
adb -d forward tcp:<port> tcp:<port>   Device
```

Danach kann der Debugger erfolgreich gestartet werden (s. Abbildung 3).

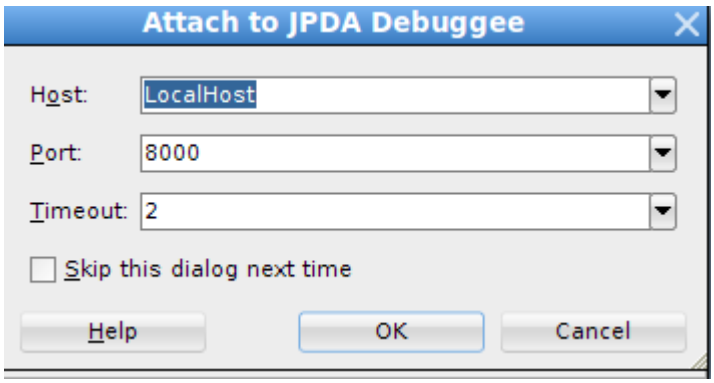

Abbildung 3: Aufruf des Remote Debuggers

Wurden in der Applikation Breakpoints gesetzt, stoppt die Applikation, sobald der erste Breakpoint erreicht wurde. Im JDeveloper bzw. OEPE können die aktuellen Belegungen der Variablen, Argumente, der Call Stack und evtl. gesetzte Watches analysiert werden. Dabei kann sich der Entwickler schrittweise durch den Code bewegen.

Sowohl für iOS als auch für Android gibt es Möglichkeiten, aktuelle Browser-Versionen für die Inspizierung laufender Applikationen zu nutzen.

Unter **iOS** kann der Safari Browser genutzt werden, um mobile Apps auf einem iOS-Gerät oder im Simulator zu inspizieren (siehe http://java.dzone.com/articles/debugging-oracle-adf-mobile).

Ab **Android 4.4** können mittels des Chrome Browsers ebenfalls mobile Apps debugged werden (siehe https://developer.chrome.com/devtools/docs/remote-debugging).

Sonstige Android Tools

Neben den Möglichkeiten, die das Mobile Application Framework (MAF) bietet, gibt es noch plattform-spezifische Werkzeuge, die den Entwickler bei der Analyse seiner Apps unterstützen.

Im Falle von **Android** ist das vor allem der Monitor des SDK (*../android-sdk-linux/tools/monitor*). Der Monitor bietet folgende Funktionalität:

- File System Explorer
- Analyse der Threads
- Analyse der Nutzung des Heap
- Netzwerk-Statistiken
- Anzeige der Log Messages.

Das bereits erwähnte Utility *../android-sdk-linux/tools/ddms* steht ebenfalls zur Verfügung, ist aber inzwischen deprecated.

Zusammenfassung

Für die Fehlersuche in Applikationen, die mit dem Oracle Mobile Application Framework (MAF) entwickelt wurden, stehen verschiedene Werkzeuge zur Verfügung. Dabei handelt es sich sowohl um Werkzeuge des Frameworks, als auch um plattform-spezifische Werkzeuge für iOS bzw. Android.
Im Rahmen des Logging ist es möglich, Protokoll-Informationen aus der Anwendung auszugeben. Mit Hilfe des Remote Debugging können Breakpoints im Java- und im Javascript-Code gesetzt werden. Anschließend kann die Anwendung im Debug Modus auf dem Gerät oder im Simulator/Emulator ausgeführt werden. Zusätzlich können plattform-spezifische Browser (Safari, Chrome) und Werkzeuge zur Problemanalyse eingesetzt werden.

Kontaktadresse:

Dr. Jürgen Menge
ORACLE Deutschland B.V. & Co. KG
Riesstr. 25
D-80992 München
Telefon: +49 (0) 89-1430-2239
E-Mail: juergen.menge@oracle.com
Internet: www.oracle.com