JavaMESupport

Using Derby on Java ME platforms

This page describes Derby's support for Java ME (also known as J2ME) environments, provides hints about how to use Derby in such environments, and links to more information.

Note: this information applies to Derby versions 10.1 to 10.10. After 10.10, the CDC/JSR 169 profile will no longer be supported. But as of 10.10, the CP2 profile of Java8 will be supported. For more information about that, see the 10.10.1.1 release notes.

- Requirements
  - Connected Device Configuration (CDC) 1.1
  - Foundation Profile (FP) 1.1
  - JSR 169: JDBC Optional Package for CDC/Foundation Profile

If your Derby version is older than version 10.1.1.0, Java ME is not supported at all. If you have a version of Derby that is 10.1.1.0 or newer, but older than 10.3.1.4, CDC/FP 1.0 is also supported. Note that there are some limitations imposed on Derby when run on a Java ME platform, see the documentation for details.

The fact that Derby supports the Foundation Profile means that other CDC profiles that are supersets of FP, for example the Personal Basis Profile (PBP), are also supported.

An overview of Java ME is available at Wikipedia.

- Documentation, Specifications and APIs
  - the Jira issue tracking the initial implementation of CDC support in Derby: DERBY-97 (this includes a functional specification)
  - the Reference Guide: JDBC Package for Connected Device Configuration/Foundation Profile (JSR169)
  - the Tools and Utilities Guide: ij.dataSource property
  - the published API: EmbeddedSimpleDataSource

(documentation links are to the alpha (development) version of the docs, see http://db.apache.org/derby/manuals/index.html for specific versions)

Related Java ME specifications and APIs (Javadoc) include:

- JSR-169 (JDBC Optional Package): API/spec (PUBLIC DRAFT) (go to the JSR's home page to obtain the final release)
- JSR-218 (CDC 1.1): CDC 1.1.2 API
- JSR-219 (FP 1.1): FP 1.1.2 API
- JSR-36 (CDC 1.0)
- JSR-46 (FP 1.0)

- How to use Derby with a Java ME VM

JDBC for Java ME (JSR-169) does not include java.sql.Driver or java.sql.DriverManager. This means that you have to use an implementation of the javax.sql.DataSource interface for accessing a Derby database on a Java ME platform. Derby (derby.jar) includes a DataSource implementation for Java ME environments: EmbeddedSimpleDataSource. Derby does not include client/server support for Java ME environments (yet).

Derby databases are portable between Java ME and Java SE platforms, provided that databases created or modified on a Java SE platform do not use features that are not supported by Derby in a Java ME environment (such as database encryption or server-side procedures).
Using Derby in Java ME applications

Derby 10.3.2 and newer include a sample application showing how to create and access a database in a Java ME environment: SimpleMobileApp. This application is available in the demo/programs/simplemobile directory of Derby -bin installations. You can also view the latest SimpleMobileApp source code from Derby's Subversion repository.

Using ij with a Java ME VM

ij is Derby's command line tool for accessing databases using JDBC. By default, ij uses the DriverManager mechanism to establish connections, which won't work in a Java ME environment. To use a DataSource instead, specify the ij.dataSource property when starting ij. In a Java ME environment this property must specify org.apache.derby.jdbc.EmbeddedSimpleDataSource.

To avoid errors on ij startup, you most likely have to include the ij.dataSource.databaseName property as well, specifying a database name, and ij.dataSource.createDatabase=create if the database does not already exist.

You can specify ij properties on the command line when starting ij (using -D<property>=<value>), or in a properties file and then use ij's -p option at startup. Property names are case sensitive.

Example property file, ij.properties:

```
ij.dataSource=org.apache.derby.jdbc.EmbeddedSimpleDataSource
ij.dataSource.databaseName=sample
ij.dataSource.createDatabase=create
```

Example startup of ij using phoneME Advanced with built-in JSR-169 support, on Linux:

```
> $CDC_HOME/bin/cvm -jar $DERBY_HOME/lib/derbyrun.jar ij -p ij.properties
```

Java ME VMs tested with Derby

Derby users and developers have reported running Derby successfully on various Virtual Machines for Java ME / CDC, including:

- phoneME Advanced
- IBM's j9

phoneME Advanced

In December 2006, Sun Microsystems released their implementation of Java ME for mobile handsets and similar devices as open source under the project name phoneME. This is licensed under the GNU General Public License, GPL v2, phoneME Feature software is addressing the mainstream feature phone segment in the mobile market, and is based on CLDC (not supported by Derby). phoneME Advanced software is addressing technical needs of devices in the advanced phone segment of this market, and is based on CDC.

You can obtain phoneME Advanced source code and binaries from the phoneME home page. To include JSR-169 (JDBC) support, you currently need to build the reference implementation of JSR-169 yourself, and link to it when building or running the phoneME Advanced VM (cvm).

Building hints are available in this blog and on the phoneME Advanced wiki. By default, phoneME does not currently include the JDBC Optional Package (JSR-169), so unless you build the VM yourself and include JSR-169, you must include the JSR-169 jar in the boot classpath of the VM, for example (for starting ij on Linux):

```
cvm -Xbootclasspath/a:jsr169.jar
   -classpath $DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
org.apache.derby.tools.ij -p ij.properties
```

(all on one line - line breaks inserted here for presentation purposes)

If JDBC support is included in the phoneME build, you may omit the bootclasspath, for example:

```
cvm
   -classpath $DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
org.apache.derby.tools.ij -p ij.properties
```

Running the Apache Derby test suite on the phoneME Advanced VM

Since phoneME b96 (r17519) it is possible to run the Derby test suite (.suites.All) on the phoneME Advanced VM(cvm) for Linux X86 without having to locally modify the phoneME source or build script. Instructions are here.
IBM’s J9 Foundation

J9 is IBM’s Virtual Machine for the Java Platform, Micro Edition (Java ME). J9 has been used in Derby testing since the dawn of Derby’s CDC support. IBM’s test reports from running Derby tests using J9 can be found here.

J9 is available on a variety of platforms through IBM’s product portfolio for small devices, and is for example part of the WebSphere Everyplace Micro Environment (WEME).

Example of starting `ij` using J9 on Windows:

```
j9 -jcl:founlll
-Xbootclasspath/a:jdbc.jar
-classpath %DERBY_HOME%/lib/derby.jar;%DERBY_HOME%/lib/derbytools.jar
org.apache.derby.tools.ij -p ij.properties
```

(all on one line - line breaks inserted here for presentation purposes)

Tips and tricks

If you run into a Derby-on-Java-ME related problem that you don’t know how to solve, post a question on the derby-user mailing list.

You may also find useful information in previous mailing list topics by searching the archives, for example Nabble.

Some specific hints are included below (feel free to add more).

Out of Memory

If you are running Derby on a resource-constrained device and frequently run out of memory, the following tips may help:

- increase the amount of memory available to the VM, if possible (for example, some VMs support the `-Xmx` option)
- tune Derby’s page cache (for example as explained here)

VM not recognized as Java ME

If the VM you are using has never been tested against Derby, you may run into new issues, such as Derby not properly recognizing your VM as a Java ME (JSR-169) environment. An indication of this may be that Derby is trying to use a class or interface that is not available on a Java ME platform, e.g. `java.sql.Driver`.

To see which Java and JDBC version Derby tries to conform with in your environment, run the `sysinfo` tool and inspect the information under the following line:

```
--------- Derby Information --------
```

It should include “J2ME” or “Java ME” and “CDC”, something like this:

```
--------- Derby Information --------
JRE - JDBC: J2ME - JDBC for CDC/FP 1.1
```

See VersionInfo for more information.

If you meet all the requirements and think you are seeing this problem, report it on the derby-user or derby-dev mailing list.