B

Classpath Extensions and User Classloaded Metadata

This section provides a description of the extensions to the `-classpath` search path and User Classloaded Metadata.

B.1 Classpath Extensions

This section provides a description of the extensions to the `-classpath` search path and `jserver` URL protocol syntaxes that allow specification of database resident objects and byte sets in search paths used by the command-line interface.

B.1.1 jserverQuotedClassPathTermPrefix

When a classpath term begins with the `jserverQuotedClassPathTermPrefix` string, it extends through the next occurrence of the string, regardless of the `File.pathSeparator` characters it may contain. The actual value of this string is given by the system property `jserver.quoted.classpath.term.prefix`. If this property is not defined, the default value is `||`.

B.1.2 jserverURLPrefix

When a dequoted classpath term begins with the `jserverURLPrefix` string, the rest of the term is treated as a URL. The value of this string is given by the system property `jserver.url.in.classpath.prefix`. If this value is `null`, any quoted term that does not begin with one of the following three prefixes, is treated as a URL:

- `jserverSpecialTokenPrefix`, if the value is set
- `JSERVER_CP`
- `JSERVER_SCHEMAc`

> **Note:**
>
> A quoted term is one that begins and ends with the string that is the value of `jserverQuotedClassPathTermPrefix`. A dequoted term is either the whole original term if it is not quoted, or the part of a quoted term between the beginning and ending occurrences of `jserverQuotedClassPathTermPrefix`.

B.1.3 jserverSpecialTokenPrefix

The value of the `jserverSpecialTokenPrefix` string is given by the system property `jserver.specialtoken.in.classpath.prefix`. If this value is not `null`, then the prefixes `JSERVER_CP` and `JSERVER_SCHEMAc` are recognized only when preceded by this string.

Related Topics

- JSERVER_CP
- JSERVER_SCHEMAc

B.1.4 JSERVER_CP

A classpath term beginning with the literal substring "`JSERVER_CP`" is converted to a URL by replacing `JSERVER_CP` with `jserver:/CP`.

B.1.5 JSERVER_SCHEMAc

A classpath term beginning with the literal substring "`JSERVER_SCHEMAc`" is converted to a URL by replacing `JSERVER_SCHEMAc` with `jserver:/CPcSCHEMAc`. Here `c` can be any character, but is typically `/`. This means that a term of the form `JSERVER_SCHEMAc + <remaining string>` is treated as a prescription for looking for shared System classloaded classes and resources in the schema identified by `<remaining string>`. For example, the term `JSERVER_SCHEMA/HR` is equivalent to `jserver:/CP/SCHEMA/HR` and it instructs to look for shared classes and resources in the schema named `HR`.

B.1.6 jserver:/CP general syntax

A URL beginning with `jserver:/CP` is meaningful only as a classpath term. The first character following `jserver:/CP` is used as the token separator for the remainder of the string. This is typically the character `/`. The subsequent tokens are the following:

- The possible values of the first token are JAR, RESOURCE, or SHARED_DATA, where RESOURCE indicates a Java resource object, SHARED_DATA indicates a Java shared data object, and JAR indicates a database resident JAR object. This token is optional and all of the values are case-insensitive. If one of these is present, the URL is called a JAR specifier. Otherwise, it is called a SCHEMA specifier.
- The value of the second token is PRIVATE. This is an optional token and is case-insensitive.
- The value of the third token is SCHEMA. This is a required token and is case-insensitive.
- The fourth token is a required token, which is interpreted as a schema name.
- The fifth token is required for a JAR specifier and prohibited for a SCHEMA specifier. It is interpreted as the name of an object in the schema identified by the fourth token, if present.

Functionally, a classpath term is used to look for an object that matches a class or resource name that is being searched for. In the case of a SCHEMA specifier, the object is looked for in the indicated schema. In the case of a JAR specifier, a particular object in the schema is identified by the fifth token in the URL. This object is treated as a JAR and the searched for object is looked for by name, within that JAR. In the case of looking for a class within a database resident JAR, this may mean finding the class as a class object in the schema. Otherwise, it means finding the search object in the actual bytes of the JAR object.

The searched for object is a database Java class object, if it meets the following conditions:

- The search name ends in `.class`
- the URL is either a SCHEMA specifier or a JAR specifier for a database resident JAR

Such a class object may be:

- Loaded as a shared system classloaded class. This is done if the optional second token PRIVATE is not present.
- Interpreted as a set of bytecodes and loaded by the `defineClass` method as a private user classloaded class. This is done if the optional second token PRIVATE is present.

> **Note:**
>
> Classes loaded from classpath terms not beginning with the jserver URL marker are always private, user classloaded classes.

B.2 User Classloaded Metadata

During database creation and upgrade, the following code is executed, which creates a table in `javavm/install/initjvma.sql`:

```
create table java$jvm$runtime$parameters (owner# number not null,flags number);
create unique index java$jvm$runtime$parameters$i on java$jvm$runtime$parameters(owner#);
```

This table is removed during downgrade by `javavm/install/rmjvm.sql`. If you want to share private class metadata and have DBA privileges, then you can populate this table manually. The rule is that if there is a row matching your owner ID, then the flag value of this row is bitwise anded with the flag value from the row with owner# = -1, if any. If none of these previously mentioned rows exist, then the bit set in the result is -1, that is, all bits set. If bit 0 (1<<0) is set in the result, then your session attempts to share existing shared private metadata. If bit 1 (1<<1) is set in the result, then the session creates shared metadata when existing shared metadata is not found. By default, there is no row in the table. So, all sessions both use and create shared private metadata.