IBM Power Systems™

Agenda Key: 407180 – 43CD
Session Number:

IBM Toolbox for Java™: Advanced

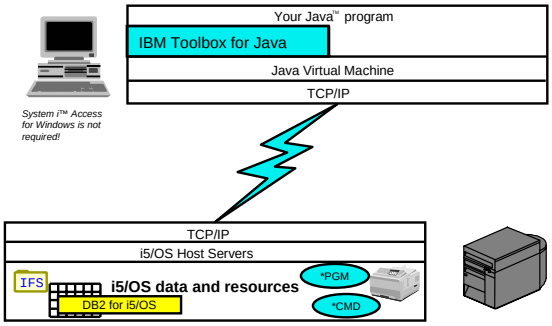
Kim Button – button@us.ibm.com

IBM Toolbox for Java™: Advanced

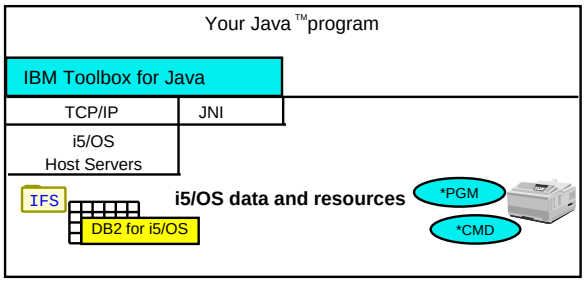
Table of Contents

IBM Toolbox for Java™: Advanced

- The Toolbox/JTOpen is a set of Java classes and utilities which provide access to i5/OS® data and resources
- Useful in client/server environments - any Java client!
  - Java client application
  - Java applet (in browser)
  - Java servlet - communicating with the i5/OS from another web server



- Toolbox runs optimized on i5/OS - makes direct API calls using JNI
  - Your application code is the same - the Toolbox selects its own implementation based on whether it is running on the i5/OS or not
- Useful in server environments - any Java server
  - Server to a client application
  - Server application
  - Java servlet - running on i5/OS



Considerations for running the Toolbox under i5/OS

API set to use:

- Native JDBC driver vs Toolbox JDBC driver
- java.io.File vs IFSFile
- Portability vs complexity
  - JNI vs ProgramCall / CommandCall

CRTJVAPGM on Toolbox file

- jt400.jar or jt400Native.jar

AS400 object can use current job's user ID and password

- When Java program and data are on the same system running i5/OS
- When Java program on one system running i5/OS and data is on another system running i5/OS

Many Toolbox components can stay in the current job using native API calls instead of a server job.

- Other functions still use server job
- CommandCall and ProgramCall do this conditionally
  - based on whether the command or program is threadsafe
  - see the setThreadSafe() method

Component list (part 1)

Component	Toolbox includes…	Native optimization
AS400 object	•	•
AS400JPing/JPing	•	
Authentication (com.ibm.as400.security.auth package)	•	•
Clustered Hashtables	•	
Command call	•	•
Connection pool	•	
Data area	•	
Data conversion	•	
Data description	•	
Data queue	•	•
Digital cerificate	•	•
Environment variable	•	
File Transfer Protocol (FTP)	•	
Graphical Toolbox (com.ibm.as400.ui.* package)	•	
GUI classes (com.ibm.as400.vaccess package) (deprecated)	•	
HTML classes (com.ibm.as400.util.html package)	•	

- Components in com.ibm.as400.access package unless otherwise noted

Component list (part 2)

Component	Toolbox includes…	Native optimization
Integrated file system (IFS)	•	Use java.io.File
JarMaker	•	
Java application call	•	
Java program information	•	
JDBC	•	Use native JDBC driver
Job and job log	•	
Message file	•	
Message queue	•	
Micro Edition classes (com.ibm.as400.micro package)	•	
NetServer	•	
Permissions	•	
Print (e.g. spooled files, printers)	•	
Product license	•	
PTF, PTFCoverLetter	•	
Product, ProductList	•	

- Components in com.ibm.as400.access package unless otherwise noted

Component	Toolbox includes…	Native optimization
Program call	•	•
Program Call Markup Language (PCML & XPCML) (com.ibm.as400.data package)	•	•
Proxy server	•	
Record-level database access	•	•
Record Format Markup Language (RFML)	•	
Resource framework (com.ibm.as400.resource package) (deprecated)	•	
Secure Sockets Layer (SSL)	•	•
Service program call	•	•
Servlet classes (com.ibm.as400.util.servlet package)	•	
Software resources	•	
System Debugger (tes.jar)	•	
System pool	•	
System status	•	
System value	•	
Toolbox installer	•	
Users and groups	•	
User Space	•	•
Validation list	•	

- Components in com.ibm.as400.access package unless otherwise noted

JTOpen (Open Source)

All of the primary Toolbox packages are open source!

http://jt400.sourceforge.net

Latest version is JTOpen 6.1

- Part of IBM's open source development community
- Use source as a debug tool
- Submit new function under the IBM Public License (IPL)
- Modify source for your use
- Submit problem reports and bug fixes

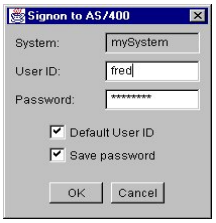
Two versions of the Toolbox:

- Licensed program (5722-JC1 or 5761-JC1)
  - Supported by IBM
  - Fixes delivered by PTFs
- Open source version
  - Supported by open source community
  - Now officially supported by IBM Service!
  - Includes source from non-IBM contributors
  - New functions and fixes available here first!

The AS400 object

Represents a connection to the i5/OS

Encapsulates security/identity



- Password caching available
- Establish a default UserID
- Sign-on GUI if UserID/password not supplied by application
- Change password GUI when appropriate
- Pluggable sign-on handler
- **Provides Secure Sockets Layer (SSL) communication**
  - Encryption and server authentication

Most Toolbox classes use the AS400 object

When running on i5/OS, Toolbox can pick up current job's user ID and password

- Use default constructor or *CURRENT

  new AS400();

  new AS400("localhost", "*CURRENT", "*CURRENT");

Represents a connection to the i5/OS

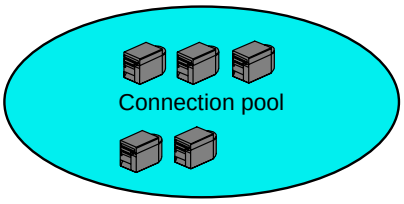
- Single vs multiple identities
- Single vs multiple connections
- Implicit vs explicit connection

```
AS400 sys  = new AS400();                    // if on client, will prompt for system, uid, pwd
AS400 sys2 = new AS400("mySystem");          // if on client, will prompt for uid, pwd
AS400 sys3 = new AS400("mySystem", "uid1", "pwd1");
AS400 sys4 = new AS400("mySystem", "uid2", "pwd2");

CommandCall cc = new CommandCall(sys);       // cc and cc2 will share a connection
CommandCall cc2 = new CommandCall(sys);
CommandCall cc3 = new CommandCall(sys3);     // cc3 and cc4 tasks will go against
CommandCall cc4 = new CommandCall(sys4);     //  different profiles
```


Connection pooling

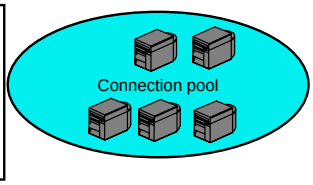
Connection pooling can improve performance

- Each new connection to the server can be an expensive operation
- Pooling means reusing AS400 objects - i.e. keeping the connection open for later
- Saves frequent disconnects and reconnects
- Common scenario: servlets
  - Without pooling: Create a new AS400 object for each invocation of the servlet
  - With pooling: Grab a preconnected AS400 object from the pool for each invocation of the servlet, return it when done!
- Connections will be added as needed



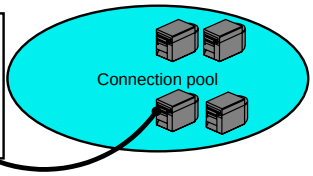
Set up the connection pool

```
AS400ConnectionPool pool = new AS400ConnectionPool();
pool.setMaxConnections(128);

// Preconnect 5 connections to the AS400.COMMAND service.
pool.fill("myAS400", "myUserID", "myPassword", AS400.COMMAND, 5);
```



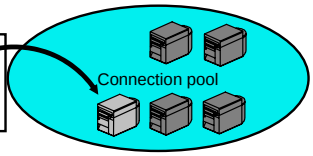
Use a connection from the pool

```
AS400 connection = pool.getConnection("myAS400", "myUserID", "myPassword",
AS400.COMMAND);

CommandCall cmd = new CommandCall(connection);
cmd.run("CRTLIB FRED");
```



Return it to the pool when done

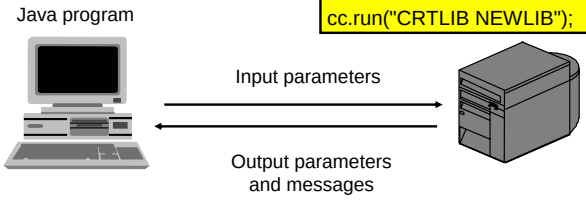
```
pool.returnConnectionToPool(connection);
```



Command call and program call

Make use of legacy code and system APIs

```
AS400 system = new AS400();
CommandCall cc = new CommandCall(system);
cc.run("CRTLIB NEWLIB");
```




```
AS400 system = new AS400();
ProgramParameter[] parmList = new ProgramParameter[n];
parmList[0] = new ProgramParameter(data);
...
ProgramCall pc = new ProgramCall(system,
                        "/QSYS.LIB/MYLIB.LIB/MYPGM.PGM", parmList);
pc.run();
```

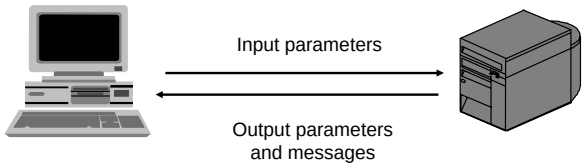
Program Call Markup Language (PCML)

Describe program calls using XML

Parameter handling in traditional Toolbox ProgramCall can be tedious.

PCML:

- Simplifies data description and conversion
- Iterative development without recompile



Traditional Program Call vs PCML

Call Retrieve User Information API using PCML

```
<pcml version="1.0">
  <struct name="usri0100">
    <data name="bytesReturned"             type="int"    length="4"  usage="output"/>
    <data name="bytesAvailable"            type="int"    length="4"  usage="output"/>
    <data name="userProfile"               type="char"   length="10" usage="output"/>
    <data name="previousSignonDate"        type="char"   length="7"  usage="output"/>
    <data name="previousSignonTime"        type="char"   length="6"  usage="output"/>
    <data                                  type="byte"   length="1"  usage="output"/>
    <data name="badSignonAttempts"         type="int"    length="4"  usage="output"/>
    <data name="status"                    type="char"   length="10" usage="output"/>
    <data name="passwordChangeDate"        type="byte"   length="8"  usage="output"/>
    <data name="noPassword"                type="char"   length="1"  usage="output"/>
    <data                                  type="byte"   length="1"  usage="output"/>
    <data name="passwordExpirationInterval" type="int"   length="4"  usage="output"/>
    <data name="datePasswordExpires"       type="byte"   length="8"  usage="output"/>
    <data name="daysUntilPasswordExpires"  type="int"    length="4"  usage="output"/>
    <data name="setPasswordToExpire"       type="char"   length="1"  usage="output"/>
    <data name="displaySignonInfo"         type="char"   length="10" usage="output"/>
  </struct>

  <program name="qsyrusri" path="/QSYS.lib/QSYRUSRI.pgm">
    <data name="receiver"       type="struct"          usage="output"   struct="usri0100"/>
    <data name="receiverLength" type="int"    length="4"  usage="input"                    />
    <data name="format"         type="char"   length="8"  usage="input"    init="USRI0100"  />
    <data name="profileName"    type="char"   length="10" usage="input"    init="*CURRENT"  />
    <data name="errorCode"      type="int"    length="4"  usage="input"    init="0"         />
  </program>
</pcml>
```

```
pcml = new ProgramCallDocument(as400System, "qsyrusri");
pcml.setValue("qsyrusri.receiverLength", new Integer((pcml.getOutputsize("qsyrusri.receiver"))));
rc = pcml.callProgram("qsyrusri");
value = pcml.getValue("qsyrusri.receiver.bytesReturned");
```

Traditional Program Call vs PCML

Call Retrieve User Information API using traditional ProgramCall

```
AS400Bin4 bin4   = new AS400Bin4();
AS400Text char6  = new AS400Text(6,  as400System);
AS400Text char7  = new AS400Text(7,  as400System);
AS400Text char8  = new AS400Text(8,  as400System);
AS400Text char10 = new AS400Text(10, as400System);

ProgramCall pc = new ProgramCall(as400System);
pc.setProgram("/QSYS.LIB/QSYRUSRI.PGM");

ProgramParameter[ ] parms = new ProgramParameter[5];

parms[0] = new ProgramParameter(100);
parms[1] = new ProgramParameter(bin4.toBytes(100));
parms[2] = new ProgramParameter(char8.toBytes("USRI0100"));
parms[3] = new ProgramParameter(char10.toBytes("*CURRENT"));
byte[ ] errorArea = new byte[32];
parms[4] = new ProgramParameter(errorArea, 32);
pc.setParameterList(parms);
pc.run();
byte[ ] data = parms[0].getOutputData();
int value = ((Integer) bin4.toObject(data, 4)).intValue();
```


Data Queues

Store data entries in a queue for processing

- Good for message passing across multiple processes
- DataQueue or KeyedDataQueue
- Supports clear, peek, read, and write operations
- Entries on queue can be ordered LIFO or FIFO
- Authority parameter useful to limit access
- Persistent

Entries are in the form of DataQueueEntry objects

- Return entry data as bytes (no data conversion)
- Return entry data as a String (converted to Unicode)
- Entry size set when queue is created (max. 64KB)

Data Queues

Example: Using a DataQueue

Process A

```
// Create a DataQueue object to represent a specific
data queue.
AS400 system = new AS400("MYSYSTEM", "MYUSERID",
"MYPASSWORD");
DataQueue dq = new DataQueue(system,
"/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

// If it doesn't exist, create it.
if (!dq.exists())
{
  dq.create(1024); // Entry length is 1KB
}

while (someCondition == true)
{
  // Wait forever until an entry appears on the queue,
then read it.
  DataQueueEntry entry = dq.read();

  // Process the entry's data.
  String information = entry.getString();

}
```

Process B

```
// Create a DataQueue object to represent a specific data
queue.
AS400 system = new AS400("MYSYSTEM", "MYUSERID",
"MYPASSWORD");
DataQueue dq = new DataQueue(system,
"/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

// If it doesn't exist, create it.
if (!dq.exists())
{
  dq.create(1024); // Entry length is 1KB
}

// Write something to the queue.
// The other process will read it.
dq.write("Some useful information.");

// When all done with the queue, delete it.
dq.delete();
```


User Spaces

Store data in an indexed memory "space"

- Good for sharing common data across multiple processes
- Supports read and write operations
- Specify offset to index inside the user space
- Set initial value and length properties
- Max. length is just under 16MB
- Authority parameter useful to limit access
- Persistent

Some i5/OS APIs return output data in a user space instead of in a ProgramCall output parameter

RFML (Record Format Markup Language)

Very similar to PCML (Program Call Markup Language)

While PCML is designed only for Program Parameters, RFML is useful for parsing/composing:

- Data queue entries
- User spaces
- Physical file records
- Data buffers

Specify record formats using XML; get/set field values

Segregate the data layout from the program logic

RFML vs. FieldDescription

Example: Composing a customer record

Using RFML:

```
import com.ibm.as400.data.RecordFormatDocument;

RecordFormatDocument rfmlDoc =
                    new RecordFormatDocument("customer");

( In a separate file named "customer.rfml": )

<rfml version="4.0" ccsid="37">
  <recordformat name="cusrec">
    <data name="cusnum" type="int"   length="2" precision="16"/>
    <data name="lstnam" type="char"  length="8"/>
    <data name="baldue" type="zoned" length="6" precision="2"/>
  </recordformat>
</rfml>
```

Without RFML:

```
import com.ibm.as400.access.AS400Text;
import com.ibm.as400.access.AS400UnsignedBin2;
import com.ibm.as400.access.AS400ZonedDecimal;
import com.ibm.as400.access.BinaryFieldDescription;
import com.ibm.as400.access.CharacterFieldDescription;
import com.ibm.as400.access.RecordFormat;
import com.ibm.as400.access.ZonedDecimalFieldDescription;

RecordFormat recFmt = new RecordFormat("cusrec");

AS400UnsignedBin2 conv1 = new AS400UnsignedBin2();
BinaryFieldDescription desc1 = new BinaryFieldDescription(conv1, "cusnum");
recFmt.addFieldDescription(desc1);

AS400Text conv2 = new AS400Text(8, 37);
CharacterFieldDescription desc2 = new CharacterFieldDescription(conv2,
"lstnam");
recFmt.addFieldDescription(desc2);

AS400ZonedDecimal conv3 = new AS400ZonedDecimal(6, 2);
ZonedDecimalFieldDescription desc3 = new
ZonedDecimalFieldDescription(conv3, "baldue");
recFmt.addFieldDescription(desc3);
```

JDBC

The Java standard for database access

Write Java programs in terms of standard JDBC interfaces, then plug in *any* JDBC driver - to work with *any* database!

- Java gives you platform independence, JDBC gives you database independence

java.sql package in Java Developers Kit

SQL is used extensively

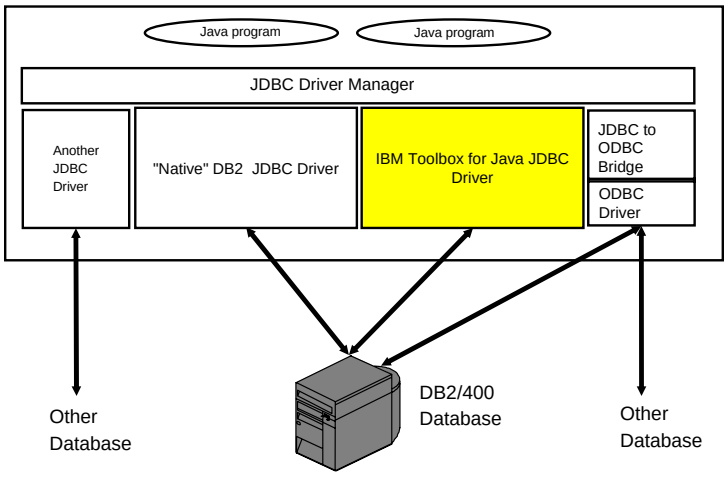
- Based on X/Open SQL Call Level Interface

Also supports:

- Database definitions, manipulations, and queries
- Stored procedures
- Catalog methods
- Transactions (commit, rollback, isolation levels, distributed)

JDBC

The Java standard for SQL database access



JDBC

Registering a JDBC driver

A JDBC driver must be registered with the DriverManager:

- **Most JDBC drivers will register themselves when they are loaded**
  - Class.forName("*JDBC.driver.class.name*"); // this is the preferred method
- **You can also register JDBC drivers explicitly**
  - DriverManager.registerDriver(new *JDBC.driver.class.name*());
- **The DriverManager can now dispatch requests to the registered JDBC driver**

```
// Register using a Java property
java   -Djdbc.drivers=com.ibm.as400.access.AS400JDBCDriver   myProgram

// Register by writing Java code
java.sql.DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());
java.sql.DriverManager.registerDriver(new com.ibm.db2.jdbc.app.DB2Driver());
```


JDBC

Connecting to a database

- **Use the DriverManager to connect to a database**
  - Connection connection = DriverManager.getConnection("jdbc:your-database's-URL");
- **Userid and password are optional**
- **The DriverManager will dispatch the connection request to the *appropriate* JDBC driver**
- **Some drivers recognize additional connection properties**

```
Properties connProps = new Properties();
connProps.put("cursor hold", "0");
connProps.put("date format", "iso");

Connection c  = DriverManager.getConnection("jdbc:as400://mySystem", connProps);
```

i5/OS JDBC driver choices

IBM Toolbox for Java JDBC driver

com.ibm.as400.access.AS400JDBCDriver

- Communicates with the database host server using TCP/IP sockets
- Provides extended dynamic performance optimizations
- Great for:
  - Client/server applications
  - Applets
  - Servlets where the Web server and data are not on the same i5/OS system

"Native" DB2 JDBC driver

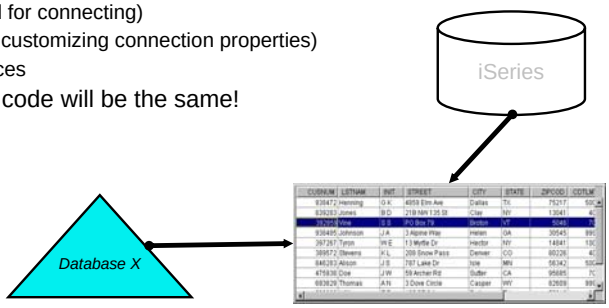
com.ibm.db2.jdbc.app.DB2Driver

- Communicates with the database using direct CLI calls
- Great for:
  - Server applications
  - Servlets where the Web server and data are on the same i5/OS system
- Toolbox JDBC driver can switch to use the DB2 driver
  - Use the JDBC property "driver=native" on the connection URL

JDBC

Code your program to be configurable

- Don't hardcode a JDBC driver
  - Allow your end users to plug in other JDBC drivers
  - Now your program works with *any* database!
- Differences between JDBC drivers:
  - Driver class name (needed for registering with the DriverManager)
  - URL syntax (needed for connecting)
  - Properties (used for customizing connection properties)
  - Subtle SQL differences
- Most of the logic and code will be the same!



JDBC

Statements

Statement "handles" are needed to issue SQL statements:

- Statement statement = connection.createStatement();
- statement.executeUpdate("INSERT INTO MYTABLE (COL1) VALUES (45)");
- ResultSet rs = statement.executeQuery("SELECT * FROM MYTABLE");

Use PreparedStatements when executing an SQL statement multiple times, or when parameters are needed:

- PreparedStatement ps =
  connection.prepareStatement("INSERT INTO MYTABLE (?)");
- ps.setInt(1, 45);
- ps.executeUpdate();

JDBC

Statements (continued)

Use CallableStatements when calling a stored procedure

```
CallableStatement cs = connection.prepareCall("CALL MYPROC (?, ?, ?)");
cs.setInt(1, 88);
cs.setInt(2, 99);
cs.registerOutParameter(2, Types.INTEGER);
cs.registerOutParameter(3, Types.VARCHAR);
cs.executeUpdate();
int n = cs.getInt(2);
String x = cs.getString(3);
```

JDBC

ResultSets

ResultSets contain the result data from a query

- ResultSet rs = statement.executeQuery("SELECT * FROM MYTABLE");
- String value = rs.getString("COLUMNA");

ResultSetMetaData objects describe the columns in a ResultSet

- ResultSetMetaData rsmd = rs.getMetaData();
- String columnName = rsmd.getColumnName(1);
- int displaySize = rsmd.getColumnDisplaySize(1);

JDBC

What else is there?

- **DatabaseMetaData**
  - Information about tables, columns, procedures, ...
- **SQLExceptions and SQLWarnings**
  - Used for error handling
- **JDBC 2.0 (only available under Java 2)**
  - Updatable, Scrollable result sets
  - LOBs (Large objects)
  - Batch updates
  - Connection Pooling
  - Data sources
  - Row sets
  - Distributed transactions
- **JDBC 3.0**
  - Savepoints
  - Parameter meta data
  - New BLOB and CLOB methods
  - Independent auxiliary storage pools (IASPs)
- **JDBC 4.0 (new in JTOpen)**
  - Automatic loading of java.sql.Driver
  - ROWID data type
  - New BLOB and CLOB methods
  - Wrapper Pattern
  - SQL/XML and XML support
  - SQLException Enhancements

IBM Toolbox for Java JDBC specifics

Connection properties

- Can be set in DriverManager.getConnection():

```
Properties connProps = new Properties();
connProps.put("cursor hold", "true");
connProps.put("date format", "iso");

Connection c  = DriverManager.getConnection("jdbc:as400://mySystem", connProps);
```

- ...or in the URL:

```
Connection c  = DriverManager.getConnection("jdbc:as400://mySystem;cursor
 hold=false;date format=iso", connProps);
```


IBM Toolbox for Java JDBC specifics

Some helpful connection properties:

Connection property	Description
"libraries"	Specify a library list, e.g. "MYLIB,*LIBL,ANOTHER"
"date format", "time format"	Specify the format for String representations of dates and times, e.g. "iso", "mdy", "usa"
"naming"	Specify the naming convention for qualified table names, either "sql" (for collection.table) or "system" (for library/file)
"block criteria", "block size"	Define block size for fetching multiple rows, can greatly improve performance
"extended dynamic",	
"package cache", etc.	Use extended dynamic support. Improves performance when same statements are prepared repeatedly - even across different runs of the program
"secure"	Use Secure Sockets Layer (SSL)
"translate binary"	Specify "true" if you have text strings stored in binary columns (some legacy programs do this)

There are many other connection properties...

Record-level database access

Fast access to i5/OS database files

Provides access to physical and logical files:

- Access records sequentially, by record number or key
- Support for locking
- Support for transactions (commit and rollback)

Options for describing the Record Format:

- The programmer can write the code
- The Toolbox can retrieve the record format at development-time and output Java source code
- The Toolbox can retrieve the record format at run-time

When running on i5/OS, direct API calls are made instead of using the host server (these are known as ***"native optimizations"***)

Record-level database access

Fast access to i5/OS database files

```
QSYSObjectPathName fileName = new QSYSObjectPathName("QIWS", "QCUSTCDT", "FILE");

SequentialFile file = new SequentialFile(as400, fileName.getPath());

file.setRecordFormat();      // Loads the record format directly from the server.

file.open();

Record data = file.readNext();

while (data != null)
{
   System.out.print((String)data.getField("INIT") + "  ");
   System.out.print((String)data.getField("LSTNAM") + "  ");
   System.out.println((BigDecimal)data.getField("BALDUE"));
   data = file.readNext();
}
```

Record-level database access

Fast access to i5/OS database files

Performance tips

- Avoid retrieving the record format multiple times. Retrieve it once and save a reference to it or hard code the record format
- Blocking factor means record caching. Experiment with different sizes or specify zero and let the Toolbox determine the blocking factor.
- Blocking factor is valid only when the file is opened for READ_ONLY or WRITE_ONLY access.
- Opening keyed files is slower than opening sequential files. Use sequential files unless you need to specifically search by key.

HTML and Servlet classes

Web components create tables and forms

Provides access to database files:

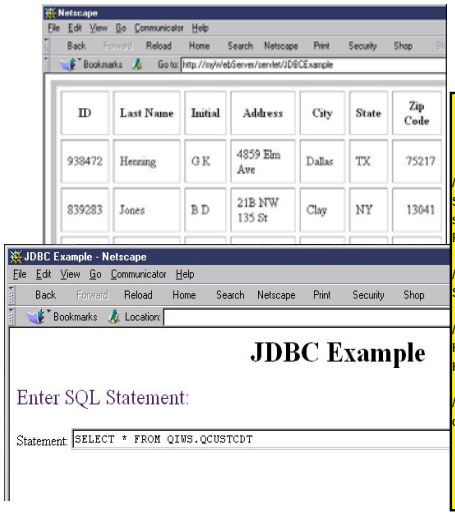
- Access database file with Record Level Access or SQL via JDBC
- Includes Meta Data

Provides classes to display data:

- Display data in tables or forms
- Toolbox provides converters that will produce HTML tables or forms based on the row data

```
HTMLTableConverter converter = new HTMLTableConverter();

ResultSet resultSet = statement.getResultSet();
SQLResultSetRowData rowdata = new SQLResultSetRowData(resultSet);

String[ ] html = converter.convert(rowdata);
out.println(html[0]);
```

Web components create tables and forms



- Classes for generating HTML output
- Useful for servlets, report generating, etc.

```
// Execute an SQL statement and get the result set.
Statement statement = connection.createStatement();
statement.execute("SELECT * FROM QIWS.QCUSTCDT");
ResultSet resultSet = statement.getResultSet();

// Create the SQLResultSetRowData object and initialize to the result set.
SQLResultSetRowData rowData = new SQLResultSetRowData(resultSet);

// Create an HTML converter object and convert the rowData to HTML.
HTMLTableConverter conv = new HTMLTableConverter();
HTMLTable[ ] html = conv.convertToTables(rowData);

// Display the HTML table generated by the converter.
out.println(html[0]);
```


HTML and Servlet classes

Web components create tree hierarchy

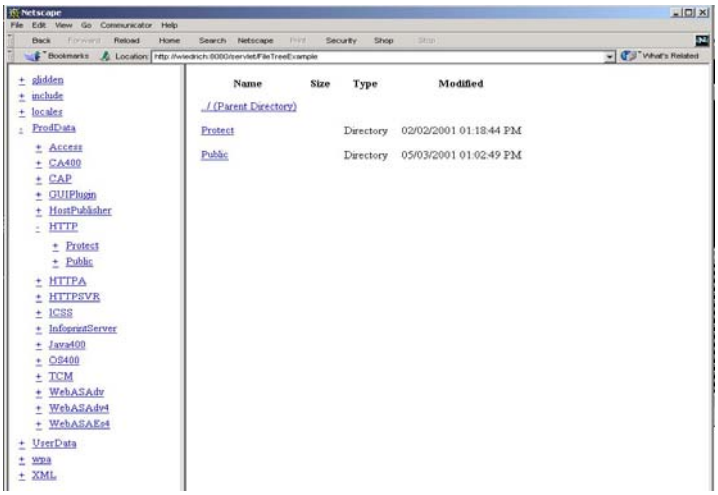
Provides classes to display the Integrated File System:

- Display contents of the Integrated File System
- Toolbox provides classes to create and display a customized and traversable tree

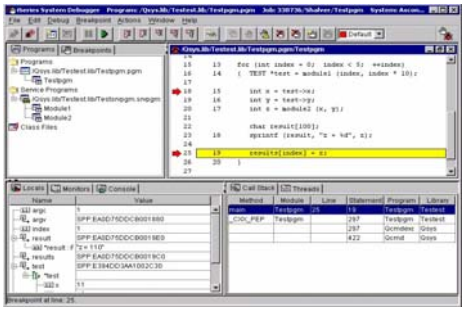
```
HTMLTree tree = new HTMLTree(HTTPrequest)

IFSJavaFile root = new IFSJavaFile(systemObject, "/QIBM");

DirFilter filter = new DirFilter();

File[] dirList = root.listFiles(filter);

for (int i=0; i<dirList.length; i++)
    {
    FileTreeElement node = new FileTreeElement(dirList[i]);
    tree.addElement(node);
    }
```

HTML and Servlet classes

Web components create tree hierarchy



System Debugger and Debug Manager



- Supports all ILE languages: C, C++, RPG, Java, Cobol, CL
- Point and click breakpoint manipulation in source code
- Automatic variable evaluation with mouse and local variable display
- Program call stack and thread display

- Requires JDK1.3 and tes.jar, jt400.jar, and jhall.jar

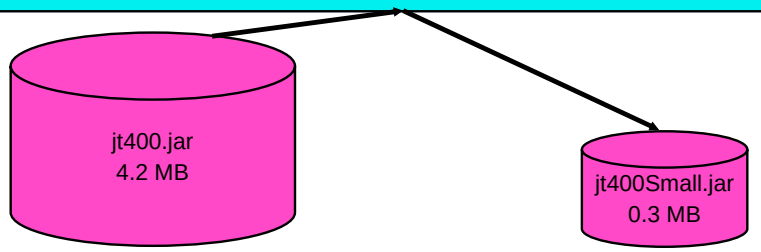
- Invoke with the following: `java utilities.DebugMgr` *or* `java utilities.Debug -s system -u user`

JarMaker

Reduce jar file sizes

- The V6R1 jt400.jar file is approximately 4.2 MB.
- A given program typically only needs a small portion of the code (e.g. only CommandCall or only JDBC).
- AS400ToolboxJarMaker pares down jt400.jar to contain only the code you need.
- JarMaker also works on jar files other than jt400.jar.

```
java utilities.AS400ToolboxJarMaker -source jt400.jar -destination jt400Small.jar -
        component CommandCall -ccsid 37 -noProxy -excludeSomeDependencies
```



References

Where can I get more information?

http://www.ibm.com/systems/i/software/toolbox/

- News, downloads, FAQs, service packs, articles, COMMON labs

http://jt400.sourceforge.net

- JTOpen - open source, bug reporting, feature requests

http://www.ibm.com/servers/eserver/support/iseries/index.html

- System i Technical Forums - including IBM Toolbox for Java/JTOpen Forum

IBM Toolbox for Java Programmers Guide

- Shipped with the IBM Toolbox for Java
- Contains overview, full API documentation (javadoc), and code examples
- Available in the i5/OS Information Center
- Link off of the Toolbox home page

Building AS/400 Client/Server Applications with Java

- Redbook SG24-2152-02

IBM Certification Testing – Here at COMMON!!

- Where and When ?
  - Delta Island F
  - 8:30 – 5:00 Tuesday and Wednesday
  - 8:30 – 12:00 Thursday
- What's in it for me ?
  - Portable credential
  - Proof that you can "Walk the Talk"
  - Peer and Employer recognition
  - Industry recognition
- How much does it cost ?
  - ***NOT*** $190 that you pay at Prometric testing centers
  - ***Special Discounted price of $95 !!***
- What tests are available?
  - System i (of course)
  - All "other" System Group platform tests (System p, x, z, and Storage)
  - All Software Group tests

Just arrived at COMMON

UPDATED

System Operator
System Administrator
ILE RPG Programmer
certification tests

See Laura Calley in the Certification Lab

Trademarks and Disclaimers