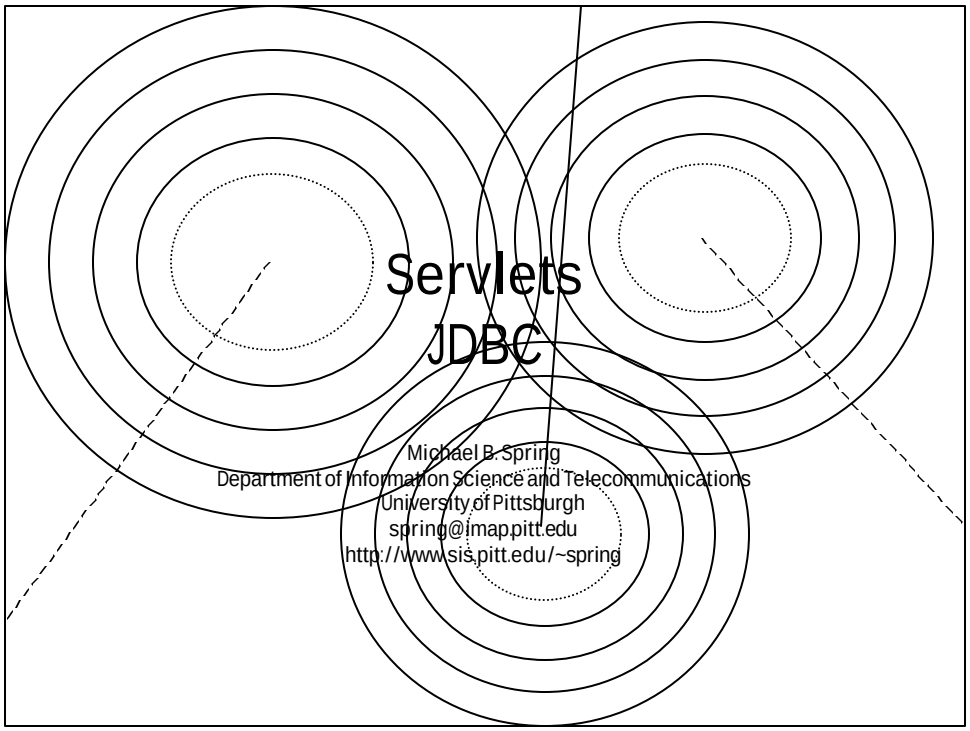

Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

A Definition

- JDBC stands for Java Database Connectivity
- JDBC is an Application Program Interface (API) that allows platform independent and vendor independent connectivity to virtually any form of proprietary relational database

SQL

- JDBC makes use of the industry standard SQL (Structured Query Language)
- In order to understand JDBC, you should have at least a fundamental understanding of SQL

Sample SQL Queries

The following query forms will be of use to those who forget the basic SQL syntax (for the final exercise)

- Select all the data in all the columns from a table:

```
SELECT * FROM table
```

- Select particular columns from a table:

```
SELECT column1, column2 FROM table
```

- Conditional select:

```
SELECT column1 FROM table WHERE column2 > 20000
```

- Perform a conditional select after a join of two tables

```
SELECT column1 FROM table1, table2 WHERE key1 = key2 AND column3 =
    'Manager'
```

- Insert a row:

```
INSERT INTO table1 VALUES (column1, column2, column3)
```

Drivers

- A vendor may support JDBC access for any given database by implementing a driver according to the JDBC driver API
- A JDBC-ODBC bridge driver is bundled with the JDK
- A list of other JDBC drivers can be found at http://java.sun.com/products/jdbc/drivers.html

Using the JDBC-ODBC Bridge with MS Access

- The example in this unit uses JDBC to perform a simple SQL query on a MS Access database
- In this example, we register the Access database with ODBC, and then use the JDBC-ODBC bridge to access the database
- It is also possible, however, to access a database through a vendor-specific JDBC driver directly

Registering the Database with ODBC

The following steps allow you to register a MS Access database with ODBC under Windows:

- from the "Start" menu, select "Start->Settings->Control Panel"
- double-click "ODBC Data Sources"
- in the tab "User DSN", select "MS Access Database", and click "Add"
- in the next dialogue box, select "Microsoft Access Driver", and click "Finish"
- enter "empDB" as the Data Source Name
- click "Select", select the file "empDB.msb", and click "OK"
- click "OK" in the "ODBC Microsoft Access Setup" dialogue box

EmpDB

- The database in our example is an MS Access database named empDB.mdb
- empDB contains a single table called empTable which looks like this:

empNumber	name	salary
1	John Doe	60000
2	Mary Jane	30000
3	Bob Smith	50000

Packages

When using JDBC, you will want to import the following packages:

- java.sql.*
- java.net.URL

Exceptions

- Many methods involved with JDBC will throw exceptions, so a significant amount of your JDBC work will be enclosed within try-catch blocks
- Most of these exceptions will be of type SQLException or one of its subclasses

Steps

The following steps are involved with making a typical database query:

- Load the driver
- Get a connection
- Execute a statement
- Parse the result set
- Close the result set, statement, and connection objects

Loading the Driver

- You load a JDBC driver by using the static method "Class.forName()"
- The argument to this method is a String containing the name of the driver you wish to use
- e.g.

```
String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
Class.forName(driver);
```

- The full name of a particular driver class is determined by the vendor

Establishing a Connection

- The next step is to create a connection using the static method "DriverManager.getConnection()"
- This method will return an object of type Connection
- e.g.

```
Connection con = DriverManager.getConnection(url, "", "");
```

- All three arguments to getConnection() are of type String. The first is a URL used to reference the database. The second and third allow for username and password
- In our example the URL is defined as ""jdbc:odbc:empDB"

Executing a Statement

- The next step is to create a Statement object using the createStatement() method of the Connection object returned in the previous step.
- A Statement object has two key methods for performing actions on a database
  - executeQuery()
  - executeUpdate()
- The arguments to executeQuery() and executeUpdate() are a String containing a valid SQL statement
- The result of the executeQuery() method is returned in an object of type ResultSet
- e.g.

```
Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery(query);
```

A Sample JDBC Connection

```
try{
// load driver
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// get connection
Connection con = DriverManager.getConnection("jdbc:odbc:empDB", "", "");
// execute statement
Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery ("SELECT * FROM empTable");
}
```


Parsing the Results

- Parsing the results will vary according to the type of queries you are making and what you want your output to look like
- In order to retrieve column values, a ResultSet object provides methods of the format getXXX(), where XXX is the desired data type
- The argument to a getXXX() method can either be a String indicating the column name or an integer indicating the column number
- The following two slides show the mappings between common Java and SQL data types as well as their associated methods

Type Mappings

SQL Type	Java Type	ResultSet Method
CHAR	String	getString()
VARCHAR	String	getString()
LONGVARCHAR	String	getString()
DATE	java.sql.Date	getDate()
TIME	java.sql.Time	getTime()
TIMESTAMP	java.sql.Timestamp	getTimeStamp()

Type Mappings

SQL Type	Java Type	ResultSet Method
NUMERIC	java.math.BigDecimal	getBigDecimal()
DECIMAL	java.math.BigDecimal	getBigDecimal()
BIT	boolean	getBoolean()
TINYINT	byte	getByte()
SMALLINT	short	getShort()
INTEGER	int	getInt()
BIGINT	long	getLong()
REAL	float	getFloat()
FLOAT	double	getDouble()
DOUBLE	double	getDouble()

Parsing the Results

- To print a meaningful result from our example, use the getString() method of the ResultSet object
- The argument to getString() is the column number
- The following lines of code iterate through each row of the ResultSet object and print out the values in columns two and three:

```
while (rs.next())
 {
     System.out.println(rs.getInt(1) + ", " +
       rs.getString(2) + ", " +
       rs.getInt(3));
 }
```

- The next() method simply moves the pointer to the next row. Note that the pointer initially points above the first row.

Parsing the Results

- The ouput of the code in the previous slide should look like this:

1, John Doe, 60000

2, Mary Jane, 30000

3, Bob Smith, 50000

Closing

- The last step is to close the result set, statement, and connection objects
- e.g.

```
rs.close();
stm.close();
con.close();
```


Recap

The following steps are involved with making a typical database query:

- Load the driver
- Get a connection
- Execute a statement
- Parse the result set
- Close the result set, statement, and connection objects

Exercise

- Modify EmpDB.java to perform more complex queries such as conditional selects, adds, and joins.
- You may wish to add more columns or rows to the given table
- If you don't have a firm background in SQL, refer to the examples shown in the slide "Sample SQL Queries"