# Database and Web App Overview

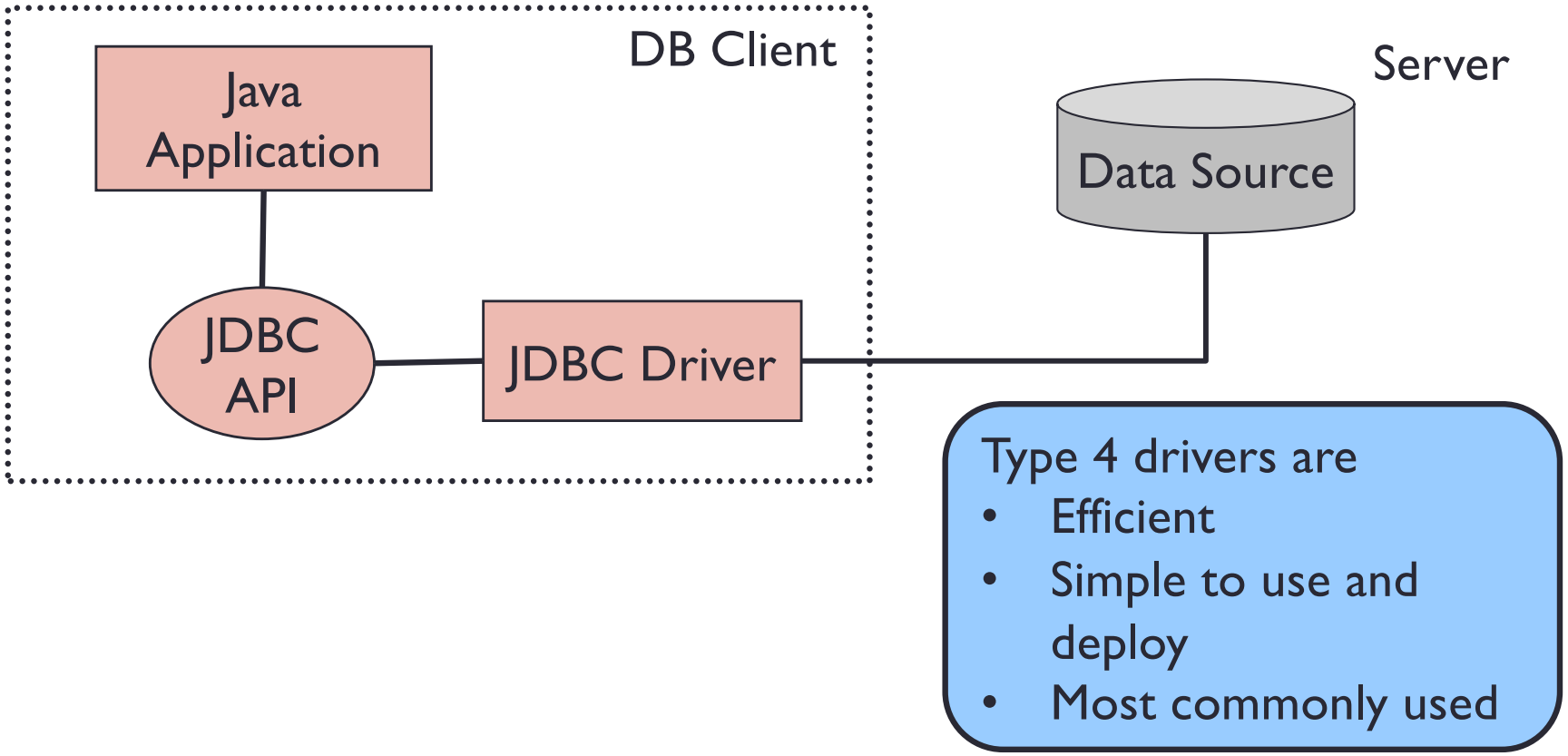
## CS 4640
## Programming Languages for Web Applications

# Java Program and Database

- JDBC API allows Java programs to connect to databases
- Database access is the same for all database vendors
- The JVM uses a JDBC driver to translate generalized JDBC calls into vendor specific database calls
- JDBC != Java Database Connectivity
  - http://www.oracle.com/technetwork/java/index.html
  - Excerpt :
    - *"JDBC (TM) is a Java (TM) API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for `Java Database Connectivity'.) "*

# Pure Java Driver (Type 4)

- There are four general types of JDBC drivers
  - We will look at Type 4



# Typical Process

1. Load the database driver
2. Obtain a connection
3. Create and execute statements
   - executeQuery – to execute SQL select statements
   - executeUpdate – to execute SQL statements that update a table
4. Use result sets to navigate through the results
5. Close the connection

# Simple Example

```
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
try
{
   Class.forName("com.mysql.jdbc.Driver");
   conn = DriverManager.getConnection("jdbc:mysql://host-name/database-name", "username", "password");

   stmt = conn.createStatement();
   registerUsers();

   stmt.close();
   conn.close();

   Driver driver = null;
   java.sql.DriverManager.deregisterDriver(driver);
} catch (SQLException se) {
   se.printStackTrace();        // handle errors for JDBC
} catch (Exception e) {
   e.printStackTrace();         // handle errors for Class.forName
} finally { // finally block used to close resources
   try {
      if (stmt != null)
         stmt.close();
   } catch (SQLException se2) {
      // nothing we can do
   }
   try
   {
      if (conn != null)
         conn.close();
   } catch (SQLException se) {
      se.printStackTrace();
   } // end finally try
} // end try
```

Load and register the database driver using ClassLoader

Connect to a database

Create SQL statement object

DB-related processes

Close SQL statement object and connection

Unregister the driver

SEVERE: The web application [/this-context] registered the JDBC driver [com.mysql.jdbc.Driver] but failed to unregister it when the web application was stopped. To prevent a memory leak, the JDBC Driver has been forcibly unregistered.

# Simple Example (select)

```
String instr_login = "some_instructor_id";
String query = "select * from user_table order by user_id, semester ;";
try
{
   ResultSet rset = stmt.executeQuery(query);
   while (rset.next())
   {
      if (rset.getString("user_id").equals(instr_login))
         instr_list.add(rset.getString("user_id"));
      else
         student_list.add(rset.getString("user_id"));
   }
   rset.close();

} catch (SQLException se) {
   se.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Prepare a query

Execute a query and store all rows retrieved in a ResultSet object

More rows?

Access a value of column "user_id"

Close a ResultSet object

By default, a ResultSet object is read-only and has a cursor that moves forward only, i.e., next().

# Simple Example (create table)

```
String query = "create table student " +
      " (student_id int, lastName varchar(255), firstName varchar(255) ) ;" ;
try
{
   int row = stmt.executeUpdate(query);
   if (row > 0)
   {
      response.getWriter().println("Table student was created.");
   }

} catch (SQLException se) {
   se.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Prepare a query

Execute a query to create a table

Successfully create a table?

# Simple Example (insert)

```
String query = "insert into student " +
        " (student_id, lastName, firstName) " +
        " values (\"4501-001\", \"Olson\", \"Mary\") ;" ;
try
{
   int row = stmt.executeUpdate(query);
   if (row > 0)
   {
      response.getWriter().println("Information was inserted.");
   }

} catch (SQLException se) {
   se.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Prepare a query

Execute a query to insert information into a table

Successfully insert information?

# Simple Example (update)

```
String query = "update table student " +
               " set lastName=\"Jefferson\" " +
               " where student_id=\"4501-001\") ;" ;
try
{
   int row = stmt.executeUpdate(query);
   if (row > 0)
   {
      response.getWriter().println("Information was updated.");
   }

} catch (SQLException se) {
   se.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Prepare a query

Execute a query to update information

Successfully update information?

By careful !!
update without a "where" clause will update **all** rows in the table

# Simple Example (delete)

```
String query = "delete from student " +
  " where student_id=\"4501-001\") ;" ;
try
{
   int row = stmt.executeUpdate(query);
   if (row > 0)
   {
      response.getWriter().println("Information was deleted.");
   }

} catch (SQLException se) {
   se.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Prepare a query

Execute a query to update information

Successfully update information?

By careful !!
delete without a "where" clause will delete **all** rows in the table

# Summary

- Most large web apps use databases to make data persistent
- The techniques for accessing databases from Java programs are identical in web apps as in stand-alone Java programs
- Read further for information on how to set up, use a database, and how to construct SQL queries