# Waukesha County Technical Institute

**152-198 Distributed Java**

# Class 6 Plan and Assignments

**Discussion Activities:**

* **Due Today and Announcements:**
  1. Maven-based “bookWebApp”, MVC-style with 2 web pages, AuthorConroller, AuthorService and Author class. Hard-code list of author objects in AuthorService shoud be returned to a report page that uses JSTL/EL to display 3 columns of info per record.
  2. IntroJDBC Sample project review.
* **Q&A**
  + IMPORTANT: don’t forget to create a branch named “NoDb” in your “bookWebApp” and use that from now on.

**Introduction to Relational Database Management Systems (RDMS)**

* Relational Database Concepts:   
  <http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm>
  + Tables, columns (fields), data types, records, relationships (joins), primary keys, foreign keys
  + Structured Query Language (SQL): <http://www.w3schools.com/sql/default.asp>
* Embedded vs. Network accessible server
  + Derby (embedded or server, standard plugin in Netbeans
    - Using the Database Management View in Netbeans (look under the “Services” tab
  + MySql (open source server): one of the most popular network RDMS in the world
    - Using the installed MySql Workbench
* You need to add a JDBC driver jar file to your “Libraries” folder in your NetBeans project.
  + Drivers for various databases can be found in the project directory for the “IntroJDBC” sample project
  + The latest MySql driver is posted on Blackboard
* **Introduction to the Java Database Connectivity API (JDBC)**
  + Java JDBC Tutorial:  
     <http://docs.oracle.com/javase/tutorial/jdbc/>
  + Netbeans Tutorial: (Connecting to Derby):   
    <https://netbeans.org/kb/docs/ide/java-db.html>
  + Netbeans Tutorial: (Installing MySql on Windows): <https://netbeans.org/kb/docs/ide/install-and-configure-mysql-server.html>
  + Netbeans Tutorial: (Connecting to MySql):  
    <https://netbeans.org/kb/docs/ide/mysql.html>
  + Instructor Demo (see screencast to be published on Blackboard)
  + Versions:
    - JDBC 2 (widely used, widely available drivers)
    - JDBC 3 (the new standard)
    - JDBC 4 (released recently; bleeding edge)
  + DriverManager class – controls loading of Java database driver software
  + Drivers: software, either native or Java-based, used to connect your Java program code to a database or middleware software
    - Type I: JDBC-ODBC Bridge (provided with JDK, requires Microsoft ODBC (Open Database Connectivity) support (least desirable, but most likely to work with any data source, including non-database soruces such as spreadsheets and text files. Use this, e.g., to connect to MS-Access databases.
    - Type II: Native API, may include some Java. Not portable because native code must be compiled specially for each platform.
    - Type III: Pure Java, but used only by Network-enabled middleware, such as a Java Enterprise Edition Application Server.
    - Type IV: the best choice – pure Java; need one for each database supported, but portable across O/S platforms
  + Other important classes in the Java API:
    - Connection class – need this to communicate with db
    - Statement class – need this to send commands to db
    - PreparedStatement class – same as above but pre-compiled and allows variables in commands (sql)
    - ResultSet class – queries produce ResultSet objects which contain the data requested; no ResultSet is produced when inserting, updating or deleting records … instead an integer is returned indicating the number of records affected.
  + Popular Relational Databases used by Java programmers (not a complete list):
    - Oracle (Commercial)
    - Microsoft SQL Server (Commercial)
    - IBM DB2 (Commercial)
    - MySQL – Free
    - PostgreSQL – Free
    - Derby -- Free
    - Others, including pure Object Databases such as Cache and db4o, and the new NoSql Databases, such as MongoDb
  + Instructor samples: (see Blackboard – “IntroJDBC.zip”. The folder contains various JDBC driver “.jar” files. However, if you ever need one you can always find them at the database vendor web sites.
  + For a **normal Java application or non-Maven** Java web app, the drivers can be installed by right-clicking on the “Libraries” folder in your project, then selecting “Add Library > Java DB”. This has already been done for you in “IntroJDBC” sample project.
  + **For a Maven-based project** you will need to add a dependency on the MySql Driveer library to your pom.xml file:

<dependencies>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-web-api</artifactId>

<version>7.0</version>

<scope>provided</scope>

</dependency>

<!-- ADD THIS -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.40</version>

</dependency>

<!-- END ADD -->

</dependencies>

* + How to install and use MySql Database and MySqlWorkbench

1. We will use MySql Database Server for the rest of the semester. MySqlWorkbench is the administrative tool.
2. Go to <http://dev.mysql.com/downloads/> to download the product, which comes in two parts: 1) the database server, and 2) the MySqlWorkbench which is the administrative tool.
3. **If you are on Windows** you can download a single installer that has both parts by first clicking on the “MySQL Community Server” link and then scrolling down about half-way on the page until you see a graphic for the installer. Notice that the graphic also describes this installer as: “All MySQL Products. For All Windows Platforms. In One Package”.   
     
     
     
     
   Now click on the Download button for the MSI installer right below the graphic. Don’t worry if you’re on a 64-bit O/S. This is the installer you want.
4. Once you’ve downloaded the installer you can run it. However, VERY IMPORTANT, don’t install if you currently have an existing version of MySQL. You must uninstall that first.
5. When running the installer accept all defaults except when it asks you whether you want to run this as a service or as an application. Choose “service”. Also, you will be asked to provide a password for the “root” user (the super user account). Please use “admin”.
6. Instructor demo
7. If you are on a **Macintosh or Linux** you will have to download the server as a separate installer, and then the MySqlWorkbench as a separate installer. Make sure you install the server FIRST!!!
8. If you are on a **Macintosh** you must not download the current version of MySqlWorkbench (v6.3.8 at the time of this writing). It has a bug that will be fixed in the upcoming v6.3.9. For now, however, you must use an earlier version 6.1.7 which is on Blackboard for your convenience.

* **A first look at writing Java code to access a database**

1. Open the “IntroJDBC” sample project in Netbeans
2. Look for the package “introjdbc”. Inside that package look for the class file “SimpleDB\_MySql\_Demo.java”
3. Compare this file with “SimpleDB\_MSSQLServer.java”, which is the MS-SQL Server version. Note that except for specific database and table references the code is basically the same.
4. These demo files are designed to keep things simple while learning – all of the code in one place, reducing complexity. However, this also violates best practices. Later we’ll learning how to use SOA to resolve this issue.

**Service Oriented Architecture (SOA)**

* + Motivations:
    - create a flexible system that can adapt to changes easily; reduce coupling between modules
    - provide interchangeable, low-level data access strategy objects that speak JDBC and are as generic as possible so the code can be used with any project. These are low-level objects in the DIP.
    - provide Data Access Objects (DAOs) that use domain terminology and translate raw data to domain objects. These are both low-level objects in the DIP and high-level objects that have data access objects as components.
    - provide Data Transfer Objects (DTOs) that use domain terminology and translate raw data from multiple columns into one pseudo-domain object. These are low-level objects in the DIP.
    - provide a Façade (service object) that hides complexity and serves as main point of access. This is a high-level object in the DIP.
  + Data transformation and challenges for flexibility (need for generic data structures in low-level db classes). How do you write code that works for any database, any table(s) and where the column names and column data types are unknown? We’ll demonstrate.
  + Time-permitting: build a simple sample in preparation for similar homework assignment

**Textbook Chapters (and other resources) covered:**

* Twitter Bootstrap: <http://getbootstrap.com/>
* CSS Tutorials: <http://w3schools.com/css>
* SQL Tutorials: <http://w3schools.com/sql>
* Java EE v1.7 tutorial: <http://docs.oracle.com/javaee/7/tutorial/doc/home.htm>
* Java SE API (v1.8): <http://docs.oracle.com/javase/8/docs/api/>
* Java EE API (v1.7): <http://docs.oracle.com/javaee/7/api/>
* Online tutorials for client-side: <http://w2schools.com>
* Netbeans web development tutorials: <https://netbeans.org/kb/trails/java-ee.html>
* Netbeans Git User Guide: <http://netbeans.org/kb/docs/ide/git.html>  
  (don’t use SSH – we’ll be using the modern HTTPS approach)

**Preparation Work for Next Class – No points for compliance, but failure to comply costs up to -10 points for each of the following:**

1. Complete any unfinished labs, including the JSTL lab discussed earlier. Make sure you push to GitHub before the start of our next class.
2. Study the JDBC Samples and continue study of the online tutorials.
3. Install MySql Database Server and MySqlWorkbench on your homework machine.
4. As part of you “bookWebApp – NoDb Branch” design a class “MySqlDBAccessor” that is flexible enough to use with any project, using any MySql Database Server (possibly at different URLs, and with different credentials) and encapsulates at least one C.R.U.D. method not discussed in class. This class will be a low-level class in the DIP.