Database Connectivity (ODBC and JDBC)

Classroom Preparation

Problem Statement

A common requirement for most of the applications that we use is that they need to maintain "persistent state". This means that certain interactions with the application have lasting effects that can be recalled hours, days, or weeks later.

- Order history at Amazon.com
- LinkedIn profile information
- Email messages in GMail

This data often needs to be searched and updated in order for the application to fulfill it's purpose. One of the most common ways an application stores persistent data is by using a database.

We've already seen how we can interact with a database directly by typing SQL commands into a GUI client (i.e. Visual Studio or PGAdmin), today we'll learn how to write application code that can interact with a database in order to read and write persistent data.

Objectives

- Review Database Design Exercise
- Making Connections
- Executing SQL Statements
- Parameterized Queries
- DAO Pattern

Notes and Examples

Review Database Design

Review Friday's database design activity with the class

Making Connections

- Application code that we write to interact with a database is a "client" of the database in the same way Visual Studio or PGAdmin is
- There are many different database vendors (e.g. PostgreSQL, SQL Server, Oracle, etc) that a Java or C# application may want to integrate with
    - Each platform (i.e. Java or .NET) provides an abstract interface (i.e. JDBC or ADO.NET) for interacting with a variety of databses in a generic way
    - A "driver" is implemented for each database so that application code can communicate with that database

- When we interact with a database, we need to **create a connection**.
    - Connections remain open until they are closed or time out.
    - Connections have overhead when created and opened, thus there is often a **finite number of connections**.

  A **connection string** specifies the name of the driver to use, the host and any port, the database name, and a username and password.

  Connection strings should not be written directly in our code. **Why?**

  Connections are valuable resources. It may not seem like a big deal if we leave it running in our single application, but what about a larger-scale application?

Executing SQL Statements

- Once a connection is instantiated and opened, it can be used by other objects that issue *SQL commands*.
- Command objects execute SQL statements and return results in the form of reader objects.
- A reader object provides the ability to walk through each row in the result set and read values from each of the columns.
- The results from a SQL query are often used to populate *domain objects*.

Parameterized Queries

A parameterized query is a query in which placeholders are used for parameters and the parameter values are supplied at execution time. The most important reason to use parameterized queries is to avoid SQL injection attacks.

DAO Pattern

The **Data Access Object (DAO)** design pattern encapsulates the details of persistent storage inside of classes whose only role is to store and retrieve data.

- **DAOs usually perform CRUD operations on *domain objects*.**
    - **C**reate
    - **R**ead
    - **U**pdate
    - **D**elete
- **DAO pattern makes code loosely coupled**
    - Isolating data access code inside of DAOs decouples the rest of the application from the details of persistence
        - Relational databases are often used for persistent storage, but other technologies could be used such as the filesystem, NoSQL database, web service, etc
        - Isolates the code changes that need to be made in the event of a table schema change.