Objectives

- Review: databases, SQL, JDBC
- Software Engineering Tools
  - Maven
- Project Organization
  - MVC

Review: Quality Attributes and Databases

- Quality attributes
  - What are quality attributes of web applications?
- Databases
  - Why are databases useful?
  - What are tables made up of?
  - What language do we use to query and update relational databases?
  - What is the syntax for the SELECT statement?
  - What is a *primary key* vs a *foreign key*?
  - How are SQL and JDBC related?
  - Compare and contrast Statements vs PreparedStatements in JDBC
  - What tips and tricks did you learn about writing/using SQL?

Understanding the Code Base

PROJECTS

What is the Ancient Graffiti Project?

Exploring AGP

http://ancientgraffiti.org/

- What is the purpose of the Ancient Graffiti project?
- What are its features?
- Who are the target users?
- What is the *vocabulary* of the project?
- What do you like? What do you not like?
- Find any bugs?
- Compare with other inscription sites for ideas and inspiration


Apache Maven Project
http://maven.apache.org/

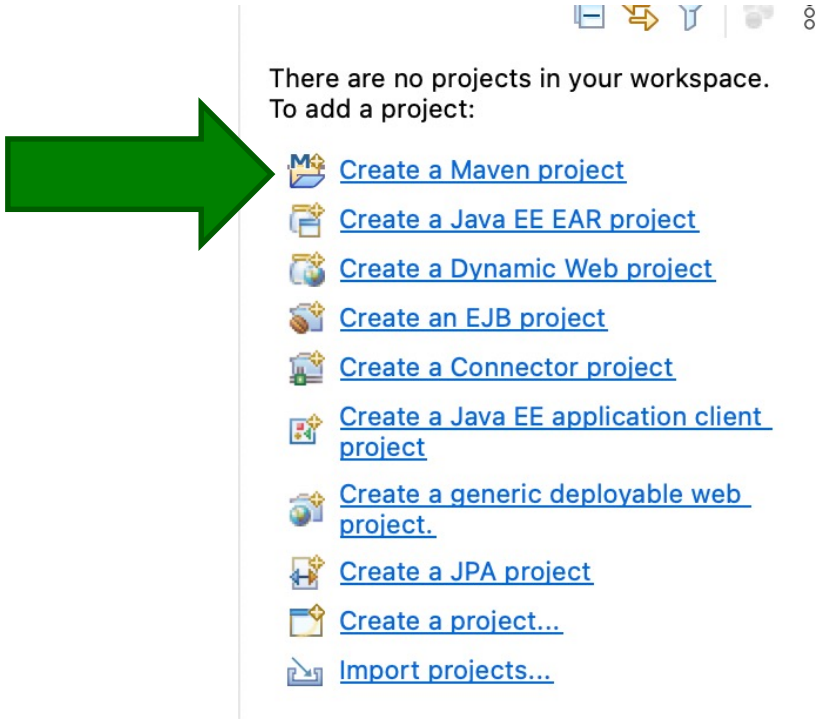
Apache Maven™

- Maven: Yiddish word meaning *accumulator of knowledge*
- Evolved from struggles in maintaining an Apache project
- For building **and managing** any Java-based project
  - Uses a Project object model (POM)
- Goal: download and build a project quickly

`http://maven.apache.org/`

Maven

- Can be used as standalone tool or within Eclipse (what we'll do)



Maven Philosophy: Convention Over Configuration

- Maven's location assumptions:
  - source code: ${basedir}/src/main/java
  - Resources: ${basedir}/src/main/resources
  - Tests: ${basedir}/src/test
- Other assumptions:
  - Produce a JAR file in ${basedir}/target
  - Compile byte code to ${basedir}/target/classes

How does this convention philosophy help us?

Maven Philosophy: Convention Over Configuration

- Benefit: reduces effort
  - Put source in the correct directory
  - Maven handles the rest
- Beyond location conventions…
- **Core plugins** apply a common set of conventions for compiling source code, packaging distributions, generating web sites, and many other processes

Consequences of Convention Over Configuration

- Users may feel forced to use a particular methodology or approach
- Most defaults can be customized
- Can create custom plugins for your requirements

Maven Build Lifecycle

- Defined by a list of *build phases*
- Example build phases
  - `compile` - compile the source code of the project
  - `test` - test the compiled source code using a suitable unit testing framework
  - `package` - take the compiled code and package it in its distributable format, such as a JAR
- When execute a phase, executes life cycle's previous phases first, in order
  - E.g., calling `package` would execute `compile` and then `test`

Maven Build Lifecycle

- 3 built-in build lifecycles
  - `default` lifecycle handles project deployment
  - `clean` lifecycle handles project cleaning
  - `site` lifecycle handles the creation of project's site documentation

Maven Repository

https://mvnrepository.com/

- Holds build artifacts and dependencies
- Typically: looking for a stable release
  - rc = release candidate (*not* what you want)

Adding a Dependency

- Several different ways dependency can be added to `pom.xml` file

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.7.1</version>
    <scope>test</scope>
</dependency>
```

- Can copy the XML code provided on the Maven repository site

Adding a Dependency

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.7.1</version>
    <scope>test</scope>
</dependency>
```

- After adding a repository, give Eclipse some time to work it out
  - Eclipse will download new dependencies
- View the Maven Dependencies in Eclipse

Updating a Dependency

- Changing versions is easy
  - Will see errors in POM if causes conflicts between dependencies
  - Will see compilation errors if upgraded version causes issues in your current code

SPRING

Spring Framework

- Open-source, powerful, and flexible framework focused on building Java applications
- Spring handles a lot of the common, “boilerplate” code so that you can focus on the logic for your application
  - Example: connecting to a database is fairly routine; just need to know which database, the user name, password, …
- Dependence on conventions over configuration
- Multiple components, e.g., WebMVC, Boot, …

Git Repositories Organization

- Our projects [will] have a private and public repository
- Configuration of projects contain private/sensitive information
  - Server location
  - User names and passwords
- Our solution
  - Keep our work private until reach a checkpoint to make public
  - Put placeholders into the config files

Software Development Review

- What is the MVC design pattern?
  - In general, how does it apply to web applications?
  - Specifically, take the Login example (from yesterday) and identify the MVC

GitHub Repository

- In the WLU-CSCI335-S24 organization
- Read the README and follow instructions

Check in on Organization

Review: How do you learn a code base?

Understanding a Code Base

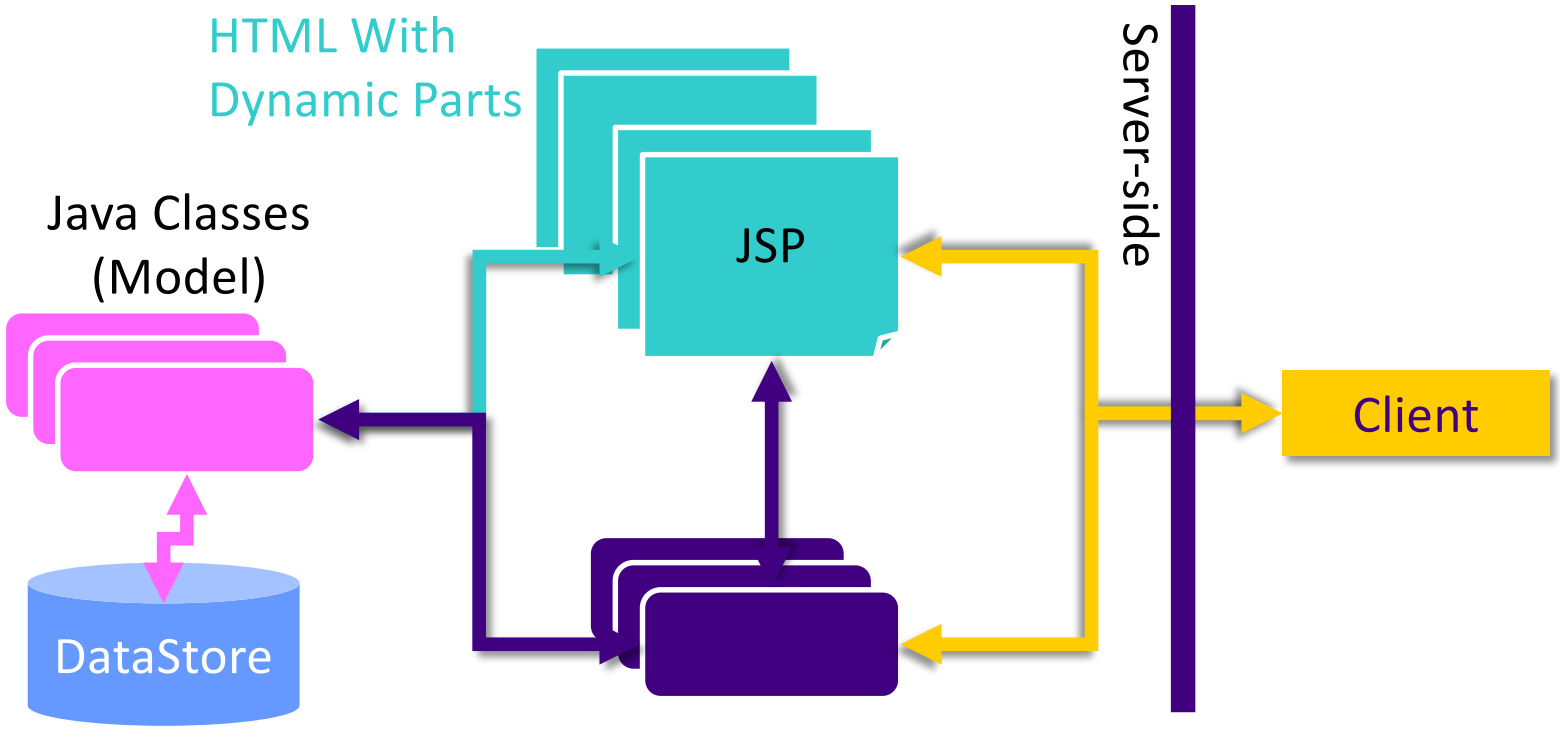
- Sooo much code!
- Find your foothold
  - Where can you find a starting point and start exploring?
- Switch between low-level and high-level
  - Low-level: looking at the files and how they belong to the big picture
  - High-level: Look in the code for something you know the application does—follow the flow

Document Your Understanding

For discussion on Monday

- In a file, document your understanding of the application
- Identify the MVC
- Label the purposes of directories
  - Where are the configuration files? What can you configure?
- Draw a picture of the architecture (similar to diagrams provided in class)
- Follow the complete flow of one use case
  - the user filters on “drawings” on the search page
  - The user selects “Food” in the featured graffiti

Web Application Architecture



Java Servlets

- Heavy lifting [code] of requests
- Forward to JSPs

Looking Ahead

- Project exploration and document – Sunday at 11:59 p.m.
- Exam on Friday