Java III Terms

Tags (in Git)

A Git tag is just a named commit. A commit is used to mark a specific point in time for a repo. As with any other commit, to continue working on a project from a tagged commit, checkout to it and create a new branch off of it to add and commit additional changes. Please note *you should have a clean working directory (i.e. everything is committed) before switching tags.*

Maven

Maven controls an application's build lifecycle (the steps involved in getting an application up and running) and can be used to pull and manage dependencies/libraries of code that need to be pulled into a project or expand on functionallity. Their are a vast amount of dependencies to choose from (string manipulation utilities, password hashing, database drivers, etc.). Dependencies may be added to your project by adding xml code to a file called pom.xml.

Tomcat

Imagine a waiter (Apache Tomcat), that listens for a customer's order (HTTP request) and then sends the order to the kitchen (the Java web application) to be processed. Tomcat then takes the processed order, the food (a response which often consists of HTML), to the custormer and continues to listen for additional requests. In this way, Tomcat works as a web server. Imagine also that this waiter, for some reason, controls the power to the cooking equitment and is responsible for turning on and off the ovens/stoves/microwaves in order for the rest of the kitchen to work with the food order. In this way, Tomcat also acts as a web/servlet container in that it creates certain conditions in the kitchen which allow the order to be processed while the kitchen is working. When the kitchen finishes running, the waiter turns back off all the cooking appliences (destroying the servlet instances).

In a production enviorment, their is one intermidiary between the waiter (Tomcat) and the customer (the client). An additional web server, Nginx will wrap Tomcat and initially recieve all HTTP requests. Imagine an assistant waiter (Nginix) that initally takes the orders and gives them to the head waiter (Tomcat). The assistant waiter is better as getting orders quickly and correctly with less risk of mixing up orders. The head waiter has a deeper knowledge of how to work with the kitchen and communicating with the kitchen. Nginx is a more efficient and secure wrapper

around Tomcat but Tomcat is needed to communicate with our Java EE application.

MVC "Model View Controller"

A way of seperating the logic/files of a web application into the three concerns of...

1. MODEL - getting or changing persistant data (often in a database)
2. VIEW - generating the HTML and front end resources to present a view to the client, often using data from the model
3. CONTROLLER - processing requests from a client and responding correctly (often by using model logic to get or change the data appropriatly and pass it to the appropriate view to send to the user)

Model (the 'M' in MVC)

In Java EE, the model logic is normally made up of several parts:

1. a model/POJO/Java Bean that represents an entity/resource that has properties that map to a the columns in a DB table, constructors, and getters and setters
2. An interface that makes a contract for how we can make manipulate (including CRUDing) the entity represented by a Java Bean (the class mentioned in point 1)
3. A DAO for a given entity (point 1) that implments the interface above from point 2 in a way that makes changes to the specific way we store the data (in our case, it will be a MySQL RDBMS). This DAO will noramlly hold an instance of a connection that allows us to use MySQL drivers to bridge the gap between Java code and MySQL changes. The JDBC API provides the MySQL drivers and a syntax communicate our SQL queries to MySQL and access reslults from our queries.

Additionally, to better secure our application, certain credentials for doing things like connecting to a database should be hidden away securely in our application. For this course, we will use a Config.java class to hold the more sensitive credentials of our app. This is a very imporant file to NEVER push to GitHub. We will include this file in a .gitignore file.

Optionally, we can create a DAO factory to create specific instances of a DAO class for a given resource and connection type.

View (the 'V' in MVC)

The view in a Java web app will normally consist of JSP files which are similar to HTML files with

additional Java code to dynamically generate the specific view a user will need at any given time in the state of our database. In a sense, JSPs are templates for HTML. For example, a view of posts in a blog on Monday may look different than the view for blog posts a month from now. The dynamic generation of HTML by Java can be done in a variety of ways but will see examples of how do this with scriplets, JSTL EL, and later in Spring, Thymeleaf.

Controller(the 'C' in MVC)

Our controller logic will mostly consist of servlets. These are special classes where we can define specific urls and listen for GET and (OR) POST requests to our application and define what to do. Often Servlets will contain logic to use a DAO to make changes to a DB or retrieve data from it, and preprepare the appropriate JSP with this data to send to the user. The view is eventually converted to plain HTML and sent to the client/user.

Servlets

See above.

JSPs "Java Server Pages"

JSPs are much like an HTML file but may contain additional Java code to dynamically generate all the HTML the user will use. Variables will often be used with a certain syntax that will evaluate to actual data (the data is often based on data in an application's database).

JSTL "JSP Standard Tag Library"

JSTL provides a specific syntax to create loops, conditional logic, and output data to create the final HTML view a that is sent to a user. Remember that JSTL is still Java server-side code and will be evaluated before it ever reaches the client/user and any client side logic that a user can interact with without needing to make another request to the application must be done client side via JavaScript.

EL "Expression Language"

EL is just the bit of syntax to output a value in the view, often used directly with JSTL.

Scriplets

Scriptlets are an older Java synatax for embedding Java logic in a view. Use of scriptlets should be avoided as much as possible to keep your JSPs as simple to edit by a non-Java programmer as possible (for example, a designer on your team).

JDBC "Java Database Connection"

An API for using database drivers to establish a connection to a DBMS and provides syntax to translate SQL queries and retrieve data from queries. Consider JDBC as an interface of methods to interact with a database and the MySQL driver we use is the implementation. By simply swapping out a few variables, using JDBC, we have the ability to use a completely different DBMS than MySQL without changing implementation details in our App.

DAO "Data Access Object"

A DAO is just an abstraction to access data. We will use a specific class that implements an interface that defines the ways an entity/resource/bean may be manipulated by our application. If we connect to a database, we will use JDBC methods in the implementation details of the interface implemented by the DAO. We will need a DAO for every resource our application handles.

Java Bean

A class used to represent a resource/entity in an application. This normally translates to a specific table with attributes for column values in the table. There are few specific requirements for how to turn a POJO (Plain Old Java Object) into a Java Bean.