**Week 13 notes**

**REST – a** server that serves up data that can be used by JavaScript and other REST clients. To do that we have a database on the back end that…

* Accepts HTTP requests

**D Beaver** – used to create tables and data

* MySQL is used to create databases and users

**Docker** – Mini isolated operating system inside of windows that runs Linux

**Behavior Driven Development (BDD) –** tests the entire app with an integration test with a process called Red Green refactor

* **Red, Green Refactor -** Write abit of thetest then run the test with JUnit. The test will fail and turns the success bar red. The status bar turns green if the test passes. Then you refactor the test until it turns red again, and you write more code to turn it green. The cycle goes on and on until the test is complete.

**TDD –** Tests methods only methods using unit test

**Test REST template** – a class that we can call methods on to send an HTTP request to app running under Tomcat. Able to test all areas of the application.

**Maven and Spring Boot**

**Maven –** a build automation tool used primarily for Java projects. Can also be used to build and manage projects written in C#, Ruby, Scala, and others. Hosted by Apache.

**Maven Dependencies –** Java archive or JAR file. This is a library with Java code that you can call into.

* MySQL connector java.jar
* Spring Web
* Lombok
* Spring Boot Dev Tools

**Extensible Markup Language (XML)** – markup language much like HTML and file format for storing, transmitting, and reconstructing arbitrary data. Defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. XML DOES NOT DO ANYTHING. It is descriptive language.

* <note>  
    <to>Tove</to>  
    <from>Jani</from>  
    <heading>Reminder</heading>  
    <body>Don't forget me this weekend!</body>  
  </note>
* This is how Maven can uniquely identify applications and tell them apart using GAV
  + G – Group ID which is domain name spelled backwards
  + A – Artifact ID is the name of the artifact you are building
  + V – Version

<**groupId**>com.promineotech</**groupId**>

<**artifactId**>jeep-sales</**artifactId**>

* + <**version**>0.0.1-SNAPSHOT</**version**>

**Dynamic Documentation**

HTTP verbs used in REST

* **Get** – retrieves one or more resources
* **Post** – Creates a new resource
* **Put** – Modifies an existing Resource
* **Delete** – Deletes a resource

Common HTTP Response Status Codes

* **200 –** OK (Success)
* **201 -** Created
* **400 –** Input is not understood
* **404 –** Resource not found
* **409 –** Conflict (Like duplicate resource)
* **500 –** Internal Server Error (Unplanned error)

HTTP and REST Compared

**HTTP**

* A protocol at the top of the internet stack
* Strictly formatted
* Understood by all internet transports
* Don’t need to worry about format cause tools and hardware will format things for you
* Doesn’t know about the payload type

**REST**

* A set of constraints on top of the HTTP protocol
* No real format – just conventions
* Only understood by the apps using it
* Need to worry about format because you are the one who creates the convention
* Payload type typically JSON
* REST knows about HTTP but HTTP doesn’t know about REST

**REST URI**

* **URI parameters**
  + Name/Value Pairs separated by = sign
  + Each pair separated by &
  + First pair starts after ?
  + Examples
    - ?model=WRANGLER&trim=Sport
    - Two pairs: model = WRANGLER, trim=Sport
* **Resource Identifiers**
  + Resource name / resource ID, or
  + Resource name / resource ID / sub resource name, or
  + Resource name / Resource ID ? URI parameters
  + Examples
    - /orders/5/options
      * Meaning: return the options in order 5
    - /orders/5/options?type=WHEEL
      * Meaning: return all wheel options in order 5

**Request / Response Payloads**

* **Typically JSON**
  + {

“model”: “WRANGLER”,

“trim”: “Sport”,

“doors”: 4,

“fuel”: HYBRID

}

**What is a Microservice?**

* A small service that does one thing like performing CRUD operations on the database
* A REST service

**OpenAPI 3 documentation**

* OpenAPI3 is a documentation standard for services
* In spring boot, we include the SpringDoc dependencies
* Annotations drive the documentation
* It’s not completely dynamic