# Spring Boot

Understands the principle of Spring Boot: It used to make rapid web app than using usual Spring Framework. The most useful part of Spring Boot is that it declare dependency in porn.xml very little and not care about dependency library version. The reason is Spring Boot have packaged individual libraries related to one larger package. These individual library have been chosen the right version so those libraries are compatible with each other. So developers do not need to care about the version of every library (which has been nightmare and takes a lot of efforts in the past). Spring Boot has solved this problem very well. It has been package dependency into common set of library such as WebMVC, Mongo Data, Test, …. In Addition, it has integrated embedded tomcat server, so build project speed has been improved help developers take less time than before.

# Spring

Spring Beans and Inversion of Control (IoC):

Understands the principle of IoC. Has practiced and learned the following:

* Declared bean in Beans.xml and use Autowired annotation in class that use that bean to initialize.
* Replaced Autowired annotation by declaring DI in Beans.xml based constructor.
* Applied declaring beans by java config (More convenient but when updating need to be compiled -> spend more time to deploy, so usually use config declared in xml for cleaner, easy tracking and not need to deploy when changes later)
* Understand how to get current ApplicationContext at anywhere in project

# Spring MVC

Understanding using some annotation in controller such as @Controller, @RequestBody, @ResponseBody, @RequestMapping. I have tried @RestController annotation which better @Controller when I don’t have to declared @ResponseBody in every method in controller. I also understands the principle of <mvc:annotation-driven /> in dispatcher servlet xml config file. I helps Spring MVC to convert JSON or XML by using appropriate library such as Jackson (JSON), JAXB (XML) and validate the model input via @Valid annotation (here I have tested with these cases). In practice webapp, I’m not using and finding out deeply about Validator Spring MVC support because I heard from OV using validator manually so I implemented my own class to validate input data in controller. One more thing in practice webapp, I have researched and used static resource config to redirect routing with static files like CSS, JS, HTML and image file (<mvc:resources />)

# MongoDb basics

Understand basics command to CRUD db, collection, document.

Understand how to config authority, permission. How to bypass authority to create, update user to grant permission to do something with some db.

# Mongo Java Driver

Understand low level java driver to interact with Mongodb. This approach is fast and full control to query and do complex tasks, but develop with it will take more efforts to do basic task than other approach.

# MongoTemplate and MongoRepository

Understand more high level java library to interact with Mongodb. This is ORM library help to map db object to class model in java. And it helps developer interact to class model, not cared much about Mongodb, so the effort to develop will decrease considerably. But this approach will execute not fast and robust customized like above approach.

In learning project app, has implemented MongoRepository to make CRUD with entities in project. And use custom query using contention name supplied by MongoRepository. Then after that, has implement another customRepo interface to use MongoTemplate to interact more customized, complex query to MongoDb.

# Maven

Understands Maven build lifecycle principle:

Maven help to build and deploy fast and consistently. It has 3 build lifecycle by order: clean, default, site

Every build lifecycle has made up from build phases. All those phases execute by order like build cycle above.

For example:

Default build lifecycle has the following build phases:

validate, compile, test, package, install, deploy

One build phase again made up of plugin goals.  
Plugin goal can be bound to zero or more build phases.  
A goal bound to more build phases will execute in all those phases called.  
Similarly, build phases have more plugin goals will execute all those goals.  
Some phases have goals bound to them by default.  
And for the default lifecycle, these bindings depend on the packaging value.

For example:

Default lifecycle binding - Packaging maven-plugin

generate-resources plugin:descriptor  
process-resources resources:resources  
compile compiler:compile   
test-compile compiler:testCompile  
test surefire:test   
install install:install  
deploy deploy:deploy

Maven have tomcat plugin which deploy webapp to embedded tomcat quite rapidly. But when use in practice, I encountered some issues. It has some drawbacks such as:

* Some error it not throws clearly relative to tomcat config
* Config as real tomcat environment not easy
* It has its own plugin for every version of tomcat. This cause some bug or incompatibility issue in latest tomcat (such as tomcat 9, it doesn’t have plugin for that version of tomcat)

Dependencies of Maven have their own scopes. Default is compiled scope if not defined. And in practice I have encountered one dependency library have “provided” scope to make tomcat work as usally. That is “javax.servlet-api”. If not defined that scope of that library, webapp cannot start because when deploy and run, it uses library of running tomcat, so it not compiled like other libraries.

# Tomcat

The first time, when I deploy webapp, I use maven tomcat plugin (it uses embedded tomcat) to deploy. It fast and straight forward. But when implementing upload file, I have encountered issue. The bug appears but it throw general error. I cannot find root cause and solution to fix. Then I switched to wtp tomcat. (Tomcat supported by Eclipse). But I have lost a quite time to config that worked. In fact, when deploy with wtp tomcat, I have to copy manually war (build from maven) to webapp folder of wtp tomcat. And it’s still not working. And I have to do some things to make it works. But I don’t like to deploy in that way. Because it quite manually and takes much time to deploy. Then I find a way on google. Then I found m2e-wtp plugin. This plugin is final and best solution as far as I know now. It makes maven and wtp tomcat of eclipse combine perfectly and very easy to deploy to wtp tomcat. It’s very fast and run like on real tomcat with all its config files. I config build automatically on eclipse. Then whenever I changes java file and saves it, It auto build and deploy on wtp tomcat right away. I don’t have to use command line “maven clean install” and copy war file to tomcat like before. And one more thing, I have tried to deploy war file to real tomcat on my local machine, it works like a charm like on wtp tomcat. So from now on, I just use m2e-wtp to build and deploy my webapp.

I have worked with some config files of tomcat such as: web.xml, server.xml, context.xml, tomcat-users.xml.

web.xml:

* Config dispatcher servlet for tomcat know which routing (servlet mapping) servlet take cares.
* Config location of applicationContext.xml file to define bean and its dependencies when initialize through context param named “contextConfigLocation”

server.xml:

* Config context path to change site url. Example: instead of go into url: localhost:8080/movieManger, just go to localhost:8080.
* Config tomcat to allow get every part from multipart data (file + json data) in client side.

context.xml:

* Use it to deploy in wtp tomcat without to reconfigure whenever add/remove project to wtp tomcat.
* Config it like in server.xml. I think in practice, developer usually config this file instead of server.xml because it applies to every webapp. Only config server.xml when something need to be applied in multiple webapps.

tomcat-users.xml:

* Config user permission and authorized info to login and manage tomcat webapp. I used it when deploy war to real tomcat by web interface.

# Uploading file

Understand how to implement upload file in server side. In fact during the implementation progress,  
I have implemented UI too. And I encountered not simple case study. Because movie data in the previous just send POST request with JSON format. Now it need to send file data by multipart form data way. So I have implemented some things in UI to send the correct request. But when UI finished, server side have have trouble too. The trouble is at that time, I used maven tomcat plugin to build and deploy war by embedded tomcat. And that tomcat has run with a bug that not cleared. I cannot find solution on internet because the error is general and when debug, it cannot attach source for me to debug native code. So I have switched to another deployment method which like real environment. That is m2e-wtp deployment in eclipse using external tomcat. When using that method, all the previous issue has been solved very simply and fast. First, the error has show concretely. I just googled one time to find the only solution. That is just config server.xml (Context element) one attribute named allowCasualMultipartParsing to true to allow get some part from multipart/form-data and enabled parsing from it. Finally it has converted and passed data to the right param of function in controller with right params I have defined. The rest is just writing file to disk and deleting file when delete or edit movie data in MongoDb. In the progress, I have also config multipart by using web.xml or by using java code. But I find more flexible when configure in java code.

# Logging

Understands how to use third party log utility to apply in practice webapp. Currently uses sl4j as abstract api to apply some log utility such as log4j, logback as concrete implementation. Log4j is easier to config (it uses yaml, better human readable). Logback use xml config and applied in OV. Understands some important and common config used in practice such as, appender, level, filter, layout format, output device such as console, file, stdout.

# Design Patterns and Refactoring

I have researched SOLID principle of OOP, some common design patterns such as Factory, Singleton, Builder pattern and some things relative to Clean code and Refactoring technique in practice.

In practice webapp, I have implemented Rest Response structure (RestResponseGeneral class) includes status, message, data field for consistency when return response to client. I refactored code when implemented this class. I have used Builder pattern in this class to initialize easier when using. I have also defined constant using Enum structure to build constant data by hierarchy for better query when needed. In controller at first, I have implemented it handling all things. Then I refactored by move all things to service for controller code to be less and cleaner. Validator class I implemented and it has been refactored too to be sure it’s not duplicated and easier to reuse in the future.

After join seminar of Lam about Clean Code topic. I apply rule in that ebook, and refactored some function return status code by using exception. This also applies Open Closed Principle of SOLID.