**My Notes for Java SpringBoot**

>>There is eclipse for java developers and eclipse for JEE, difference is a few features like webtools, servertools, jpa, jsf. These features can be added by installing webtools plugin in normal java eclipse.

>>Maven is a build management tool for spring/springboot projects.

>>Backing up project-

Right click on project->export->general->archive->save in zip->compress the content of the files->finish

Import->Maven->Existing Maven Project

>>To make a java web app spring, spring mvc

>>Use of Spring requires these to be defined:

* Component scan
* View resolver- for web app dev
* Data source- for database related app
* Non-functional requirements:

1. Logging- log4j
2. Error handling
3. Monitoring
4. Auto configuration

* Dependency management- done inside pom.xml file
* Web app configuration inside web.xml
* Manage beans in content.xml

>>Starting a spring boot project:

1. Go to start.spring.io and create a starter project and download & extract it.
2. Open eclipse and import the project under existing maven project category

**Some popular Dependencies:**

1. SpringBoot Starter Web dependency is needed for RESTful api, also provides the auto configuration for web app developement
2. SpringBoot Unit Test- for unit testing
3. Spring JPA- For Database connectivity to Mysql/oracle database

**Configuration:**

Spring boot comes with auto-configure.JAR containing different java packages with diff preset configurations by default. This is known as auto configuration.

Auto configuration happens based on JAR dependencies added to project

Manual configuration can be done from within Application.properties file.

You can also make changes to java packages within Auto-configure.jar for custom manual configuration.

What are auto configured: Default logging-info, default server-tomcat, default error page, default JSON conversion(bean to JSON)..etc

**The Application.properties**

* used for JDBC connection
* For manual configuration
* For activating particular profiles

>>SpringBoot production readiness

Logging- Log4j

Testing- Selenium,TestNG, Junit

Monitoring- Spring Boot Actuator

>>Spring Boot DevTools- It’s depency that when added to spring boot project will automatically reload the web app after every change in app code.

**Profiles**

Profiles allow you to define different configurations for same app depending on the environment. For e.g. Dev, Test, Stage, Prod.

To do this all you need to do is copy paste the application.properties file in the same location by renaming it like

Application-dev.properties = for dev enviroment

Application-prod.properties = for prod environment

.

.

.

So on and so forth

>> To activate one particular configuration properties (for e.g Application-dev.properties) write the following command in application.properties file

Spring.profiles.active=dev

>>Unless a profile is activated in the application.properties file only the default autoconfiguration or manual configuration application.properties will run.

Example:

application.properties

server.port=8082

logging.level.org.springframework=info

spring.profiles.active=dev

application-dev.properties

server.port=8082

logging.level.org.springframework=trace

In above example logging level will trace when dev profile is activated, otherwise comment out the 3rd line in application.properties file the it will be info

Debug levels:

Trace

Debug

Info

Error

Off

**Annotations:**

>>Annotations are special keywords that are used to provide specific instructions to compiler. Could be used to for additional functionality or control flow of execution (like for testcases in testing). Every java framework comes with it’s own set of annotations.

>>Some important annotations

1. @RestController- Lets the compiler know that methods to be exposed are here within this class
2. @RequestMapping()- specific parts of the program can be exposed to end user throught postman or browser.
3. @Component- auto creates an instance of the class
4. @AutoWired- allows to use the autocreated class instance
5. @ConfiguratioProperties(prefix-“currency”)- allows you to use values defined in application.properties file within java classes(depending on the active profile) for class variables. For.eg. currency.url, currency-key

>>@RestController & @RequestMapping are part SpringMVC

**Spring-Actuator**

>>Monitor and manage your application

>>Get info of your app, e.g. env, metrics

>>Provides no of endpoints:

* Beans- Compete list of spring beans in the app
* Health- Application health information
* Metric- Application metrics
* Mappings- Details around Request Mappings

>>The actuator tool can be integrated in the app using the dependency

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

>>The actuator can be accessed using <http://localhost:8080/actuator>

>>By default only health endpoint is exposed, additional endpoints need to be exposed by configuring application.properties file. For e.g.

management.endpoints.web.exposure.include=\*

**Spring vs SpringBoot vs SpringMVC**

>>Spring- @Component @Autowired, Dependency injection

>>MVC- @RestController, @RequestMApping

>>SpringBoot- Autconfiguration, Starter project, Embeded Server, Default logging, Actuator, Profiles & ConfigurationProperties

What is an API anyway?

API(Application programming interface), basically it’s a mechanism that allows two programs to interact.

For e.g. Suppose you are building a weather app, and google already has a program that collects local weather data, now you can integrate an API in your app which allows your app to communicate with google’s program and get the weather data for a particular location.

In a way APIs are similar to header files, difference instead of using precoded static code we are using a live running program in our code.

REST API?

Our rest api is basically a java program running on local embedded server, which the end user can interact with using the port in the browser

Steps to make a REST API(Very Easy):

1. Go to start.spring.io and create a starter project and download & extract it. Project type=Maven, Language=Java, Dependency= Spring Web+etc, Java= Java 11, Artifact= The name for your project
2. Open eclipse and import the project under existing maven project category
3. The base package in the project will contain 1 java class, which will the driver class, don’t tamper with this class.
4. Add another class which will implement the Rest annotations.
5. This class will contain method/methods that we will expose as endpoints using @RequestMapping(“/endpoint\_label “) annotation.
6. The class will have @RestController annotation to let the compiler know that this is the rest controller class and this class will create the methods with @RequestMApping annotations in it.
7. The methods must return something, and that will be the output you will get upon accessing the function from the browser using endpoint label

>>Getters refer to class methods to that take values for class variables.

>>toString() method is used for String objects(not String variables), this returns the string representation of the object

**public** **class** Geeks {

**public** **static** **void** main(String args[])

    {

        String Strobj = **new** String("Welcome to the world of geeks.");

        System.out.print("Output String Value :");

        System.out.println(Strobj.toString());

        String Strobj2 = **new** String("Let's make it simple for you.");

        System.out.print("Output String Value :");

        System.out.println(Strobj2.toString());

    }

}

>>How to upgrade existing eclipse, refer the below link:

<https://wiki.eclipse.org/FAQ_How_do_I_upgrade_Eclipse_IDE%3F>

>>Our Rest api will display info in JSON format

What you did as java developer in previous company?

I did automation testing in java for about 5 months then for about 7 months I was in java support role there my responsibilities was to take care of tickets analyze code for error and then further raise the bugs in jira