Full Stack --

2 Sections --- Development Section (Designing, Coding, Testing) & Operations Section (Deploying, Automating, Maintaining)

3 Layers – Front End (User Interface/View) , BackEnd (Business Layer/Service Layer/Server Side Code), Database Layer

Front End – HTML/CSS/JS or JS Based Framework (Angular, React, Vue etc.,) -- Client Side – Web based needs a web browser (Which create and sends the request to the server)

BackEnd – Java, SpringMVC, SpringBoot, JSP, Servlet, JSF, Struts

Database – JDBC, JPA, ORM (Hibernate, EclipseLink,iBatis), OXM, JMS [ SQL based / No-SQL based (MongoDB, Redis, Neo4J, Cassandra …. )]

Client – Server Model – It used Request & Response Object

Example : Restaurant Food Order

HTML – We can able to create Static Web Pages (The contents will not change with respect to user/time)

Registration Form

Login Form,

Displaying the Static Data, showing images in many cards.

HTML alone can’t be used to create dynamic content

To create dynamic content we need to use any server side code.

We can’t run a for loop, we can’t execute condition checking.

Validation --- Checking validity of the data/ user input

Login Form --- Username, password (fields) – username should have minimum 8 characters, no special symbols and spaces are allowed. Only alpha numeric data allowed.

Validation – checking the data for correctness and satisfying the given conditions.

Two types of validation

1. Client Side Validation
2. Server Side Validation

HTML- Static Content

Dynamic Content – JSP (HTML+Java) , Servlet (JAVA+html) – ASP, PHP --- (Server Side Code) – it needs a web server to run (tomcat, glass fish, weblogic, wildfly)

Jsp tags <% %>, <%! %>, <%@ %>, <jsp:useBean> </jsp:useBean>

Web Service –

1. SOAP based – Simple Object Access Protocol (XML, WSDL – WebService Definition/Description Lang)
2. REST based – Representational State Transfer. – (Reuse the HTTP methods)

HTTP Methods

1. Get - Getting the data from server (Read)
2. Post - Sending the data to the server to create new resource (Create/Insert)
3. Put - Sending the data to the server to modify existing resource (Update)
4. Delete – Sending data to server to remove existing resource (Delete)
5. Options
6. Trace

Creating REST based Web Service –

Spring MVC – Spring Boot Web Service

|  |  |  |
| --- | --- | --- |
|  | Spring MVC | Spring Boot Web Service |
| 1 | It’s a Module in Spring Framework | It’s a way of creating Spring Application |
| 2 | It generate HTML responses | It generates JSON response |
| 3 | @Controller annotation is used | @RestController annotation is used |
| 4 | ModelAndView object is used | @RequestBody to create Model object from the Request |

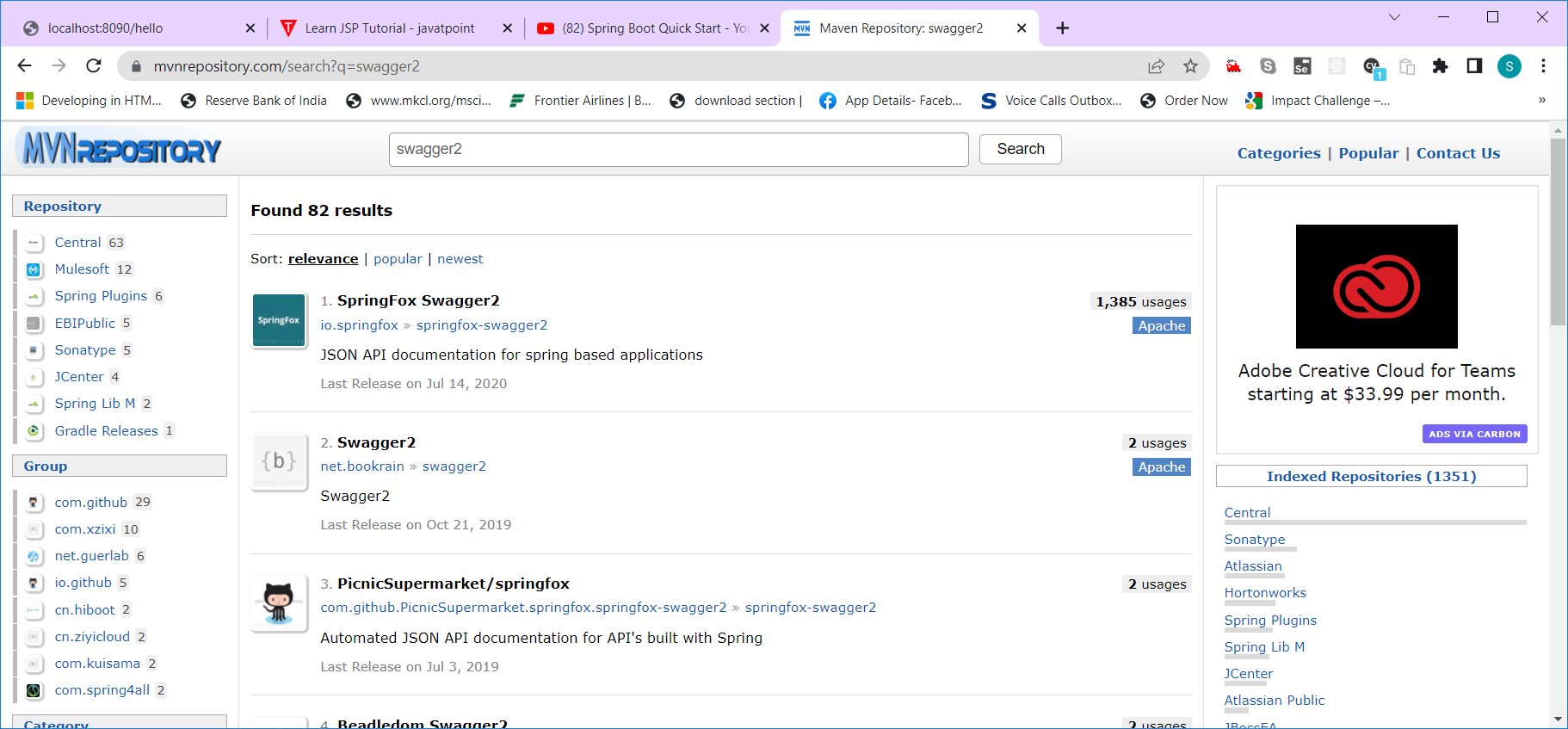
Swagger --- API Documentation (External Dependency)

Web Service – Swagger is a library to generate API Documentation (Web Service Manual)

This will display all the end points, and it’s usage information.

Washing Machine --- User Manual (How to operate – How to resolve problems, usage guide, safety tips)

Mvnrepository.com



<dependency>

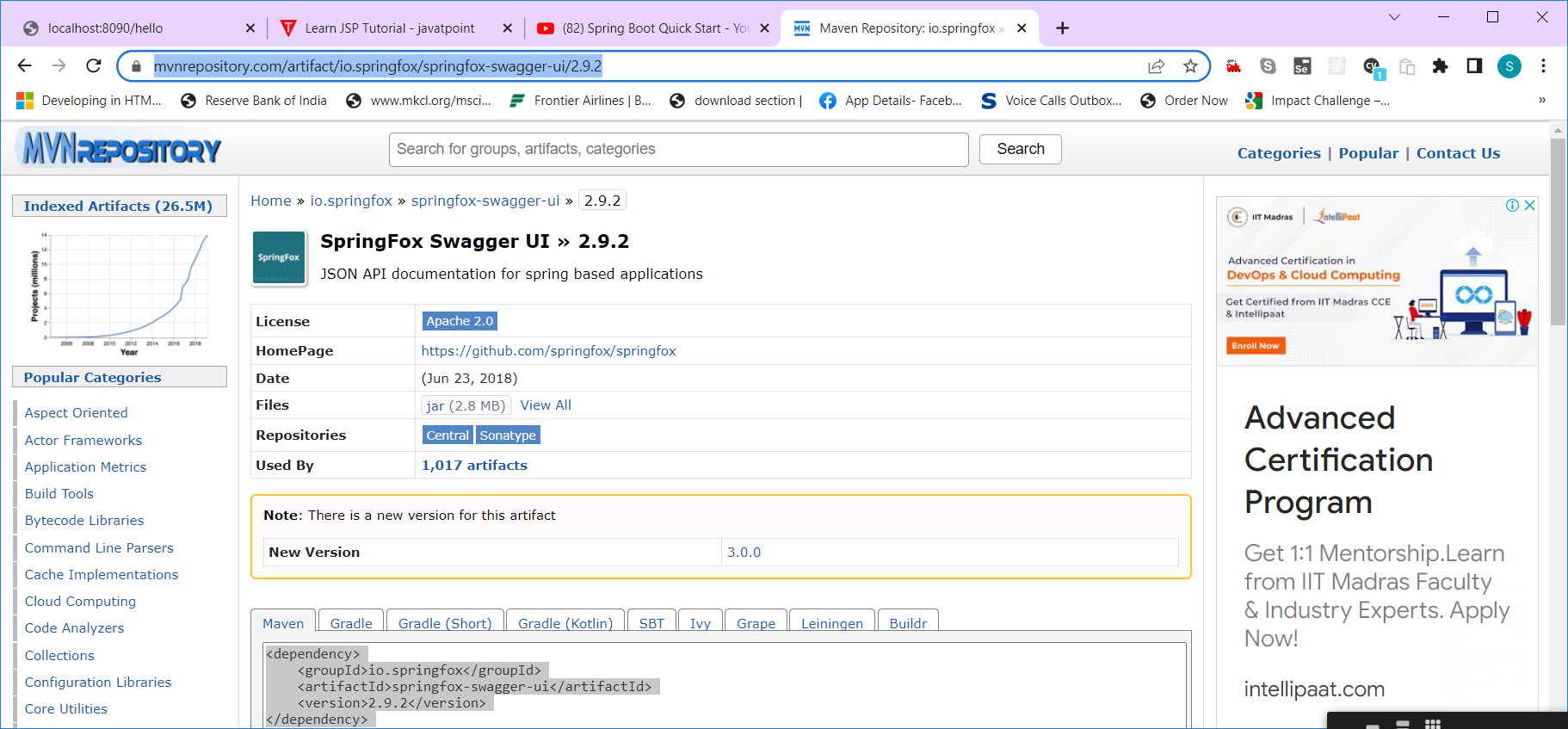
<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui/2.9.2>



<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

Note: Use Spring Boot Version 2.4.2

Add @EnableSwagger2 annotation to the main method of the starter class

@SpringBootApplication

@EnableSwagger2

**public** **class** MyserviceApplication {

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

SpringApplication.*run*(MyserviceApplication.**class**, args);

}

}

Two end points will be added automatically (localhost:8090/v2/api-docs, localhost:8090/swagger-ui.html)

