Day 34 Agenda

Spring Boot, Actuator, Dev Tools, RestTemplate to access Web Service, Spring Data JPA

Day 33 Revisit

Spring framework – It’s a Java based framework used to create Loosely coupled Java Applications.

It contains many modules 1)Core 2) Data 3) Security 4) Web 5) Test

Spring uses two important design patterns 1) IoC (Inversion of Control – Using own car or ola cab) 2)

DI (Dependency Injection) [Injecting medicines directly to the blood for quick recovery]

Though Spring is a very popular java based framework, it used xml based configuration file namely

applicationContext.xml

spring.xml

XML is case and space sensitive.

Any Changes in xml file needs complete restart of the server and complete application re-build & re-deployment is needed.

To Resolve the challenges of the Spring framework, Spring framework developers released a new updated version of spring which is called as Spring Boot.

Spring Boot is a very popular Java based framework for creating any type of Java based applications.

Spring Boot is mainly used to Create JVM based Web Services & Micro Services

Spring Boot supports JAR & WAR packaging. It also supports JAVA, Kotlin, Groovy (JVM based languages)

Springboot supports both Maven & Gradle project.

SpringBoot project will contain pom.xml only, for other configurations instead of xml file it uses text based configuration file namely application.properties

Spring boot official URL is <https://spring.io>

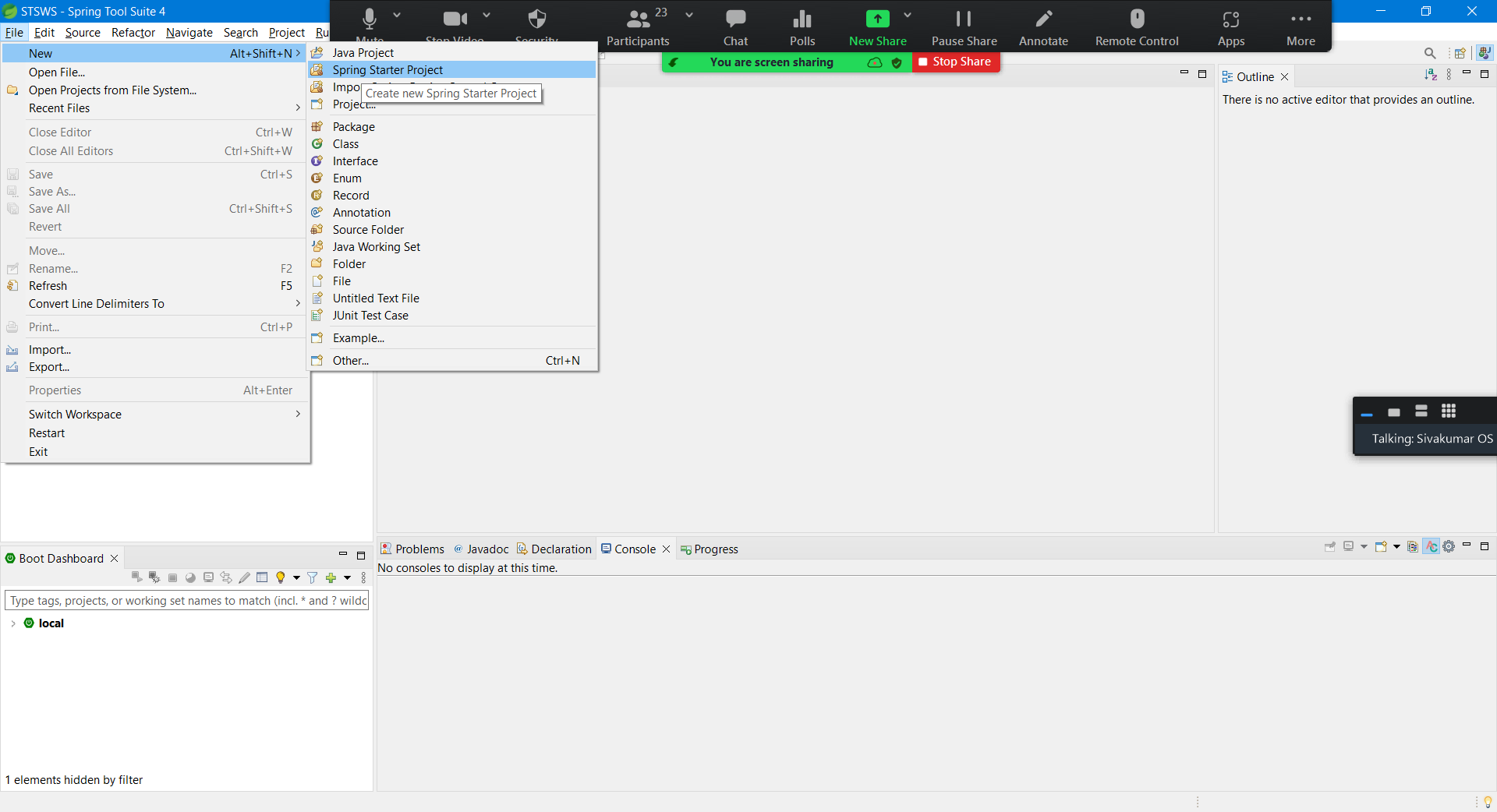
3 ways of Creating a Spring Boot Application ( Web Services or Micro Services)

1. Using Spring Initializr (<https://start.spring.io> )
2. Using STS (Eclipse based IDE) – Spring Tools Suite
3. Using Spring Boot CLI (Command Line Interface)

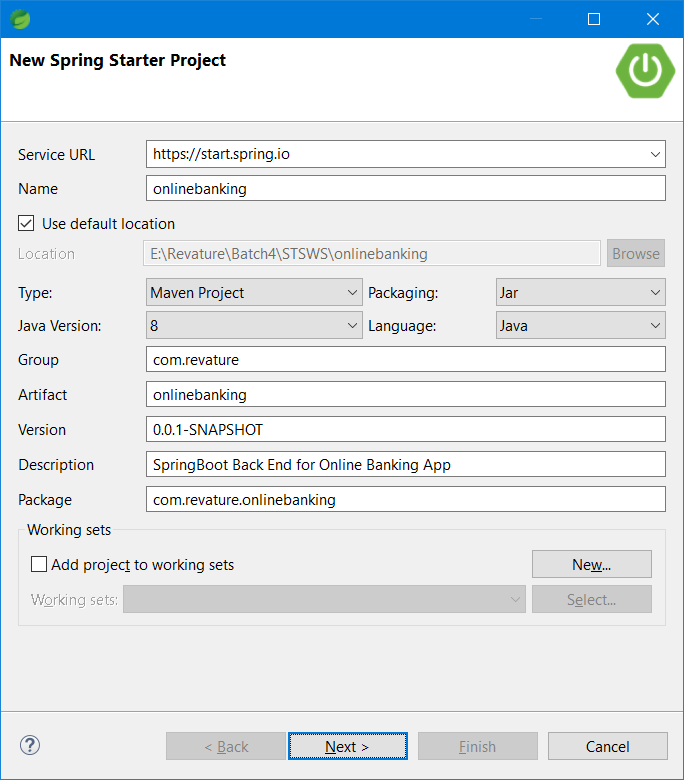
IDE – Integrated Development Environment

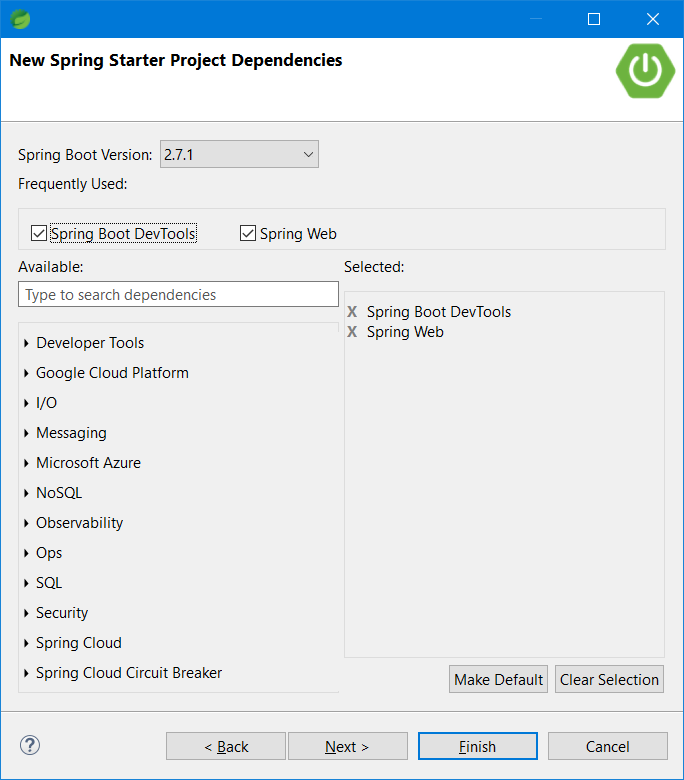
Steps to Create a New Spring Boot Project

* Open Eclipse/STS
* Create a new “Spring Starter Project” (File🡪 New🡪 Spring Starter Project)



Fill the details of the project





* Click on Finish button

Project 2 Dependencies

1. Spring Web
2. SpringBoot Dev Tools
3. Spring Data JPA (for DB CRUD operation)
4. MySQL/Postgres/H2 (In-memory configuration)
5. API Documentation
6. Actuators,Swagger (API Documentation tool)

Project Package structure

Com.revature.<proj\_name> (base package)

Com.revature.<proj\_name>.controller (All the controller classes)

Com.revature.<proj\_name>.model (All the Entity Bean Classes)

Com.revature.<proj\_name>.repo (All the DAO interfaces & Classes)

Com.revature.<proj\_name>.service (All the service Interface & Classes)

Com.revature.<proj\_name>.exception (All the Custom Exception classes)

Com.revature.<proj\_name>.util (All Utility and helper classes)

CREATE SCHEMA `online\_banking` ; (MySQL/Postgres/H2)

Controller class is controlling both data flow and control flow. [Data will flow from DB Server to Client – Read Operation], Data flow from User (Client) to the DB Server – Write Operation

Controller class is the place where we will map the URL using http methods. Also this is the place where all the end-points of the web-service applications are defined.

One Customer can have multiple accounts [One to Many Relationship]

Transaction

Id, datetime, amount, type(deposit, withdraw, transfer), accNumber, beneficiary

SpringBoot

1. Entity Bean class (@Entity, @Id, - Must have annotations in each entity bean class) [Optional annotations - @Table, @Column, @NotNull, @Max]
2. Repository Interface that needs to extends JpaRepository<T,ID>
3. ServiceInterface and Impl class for the entity bean class
4. Controller (@RestController, @GetMapping, @PutMapping, @PostMapping, @DeleteMapping, @PathVariable, @RequestBody)

<https://www.javaguides.net/2019/08/spring-boot-jpa-hibernate-one-to-many-example-tutorial.html>

<https://www.callicoder.com/hibernate-spring-boot-jpa-one-to-many-mapping-example/>