

Software Engineering for the WWW

Spring Boot

Dr. Vinod Dubey

SWE 642

George Mason University

Spring Boot

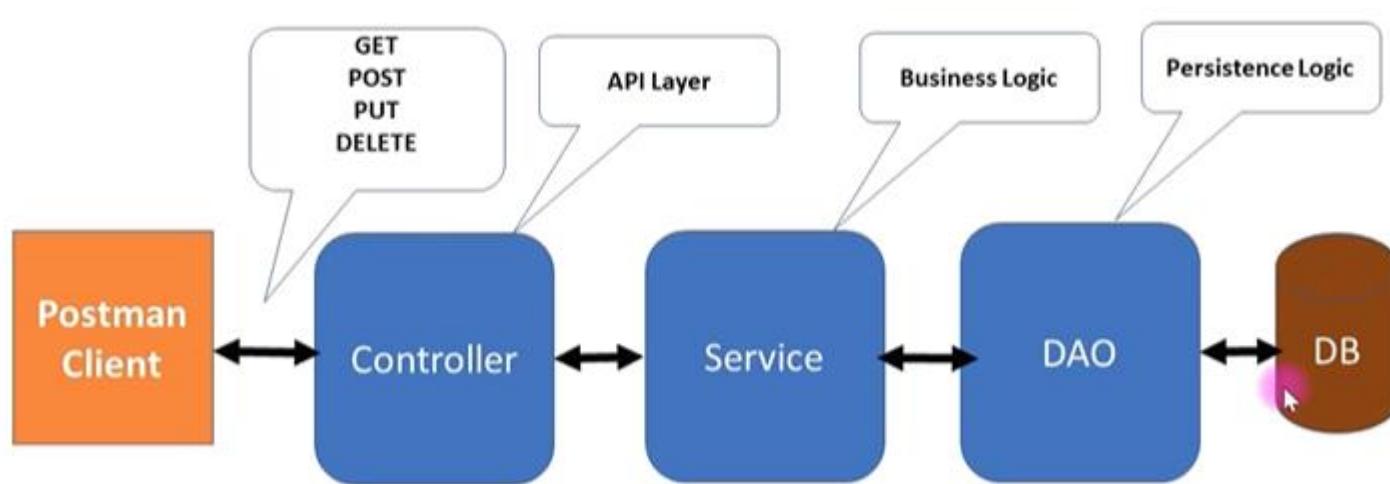
- **Agenda**
 - Installing JDK
 - Spring Boot
 - Creating RESTful and JPA application using Spring Boot
 - Installing MySQL server and Workbench

Spring Boot - Goal

- **How to implement REST APIs using**
 - Spring Boot,
 - Spring Data JPA (Hibernate), and
 - MySQL

Spring Boot Tutorial

- Typical three-layer architecture to implement Spring Boot based applications
 - Controller layer hosts REST APIs
 - Service layer encapsulates business logic, and
 - DAO that encapsulates data persistence logic.
 - Postman - REST client



Source: youtube

Spring Boot Tutorial

- **Student Management System that allows REST client to**
 - To get list of students
 - Get a single student by id
 - Create a new student
 - Update an existing student
 - Delete a student

Spring Boot Tutorial

- REST APIs for Student resource

HTTP Method	URL Path	Status Code	Description
GET	/api/students	200 (OK)	Get all students
GET	/api/students/{id}	200 (OK)	Get single student by Id
POST	/api/student	201 (Created)	Create a new student
PUT	/api/students/{id}	200 (OK)	Update an existing student with id
DELETE	/api/students/{id}	200 (OK)	Delete a student with Id

Spring Boot Tutorial

- **Tools and technologies used**
 - JDK 17
 - Spring Boot 2.5.0
 - Spring Data JPA, which internally uses Hibernate as the default JPA provider
 - Embedded Tomcat server 8.5+
 - MySQL database
 - Maven 3.2+
 - Eclipse STS

Spring Boot Tutorial

- **Creating STS (Spring Tool Suite) project**
 - STS IDE requires latest version of Java, e.g., JDK17 preinstalled
 - Download latest version of **JDK** and install on the machine where you will run the STS IDE
 - Download **STS IDE** as a jar file, extract, and **launch STS IDE** on local machine
 - Create a Spring Boot application

Spring Tool Suite requires Java JDK

- Download and install JDK

A screenshot of a Google search results page. The search bar at the top contains the query "jdk download". Below the search bar, there are several navigation links: All (selected), Books, Videos, News, Shopping, More, and Tools. The search results section shows the following information:

About 2,860,000,000 results (0.48 seconds)

<https://www.oracle.com/java/technologies/downloads.html> ::

Java Downloads | Oracle ✓

Java SE Development Kit 17.0.1 **downloads**. Thank you for **downloading** this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a ...

[1 JDK 8 and JRE 8 Installation...](#) · [JDK 11 · Java EE](#)

<https://www.oracle.com/java/technologies/jdk17-archive-downloads.html> ::

Java SE 17 Archive Downloads - Oracle ✓

Java SE 17 Archive Downloads. Go to the Oracle Java Archive page.

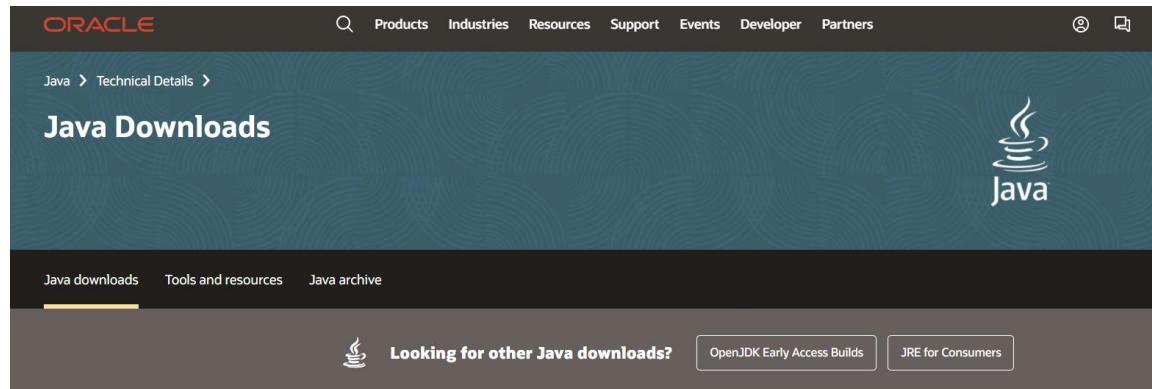
<https://www.oracle.com/technologies/java-se-glance.html> ::

Java SE | Oracle Technology Network ✓

Java Platform lets you develop and deploy **Java** applications on desktops and servers, as well as in today's ... [Download](#) · [Release Notes](#) · [Press Release](#) ...

Spring Tool Suite requires Java JDK

- Download and install JDK



Java 17 available now

Java 17 LTS is the latest long-term support release for the Java SE platform. JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions](#).

[Learn about Java SE Subscription](#)

JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17.0.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

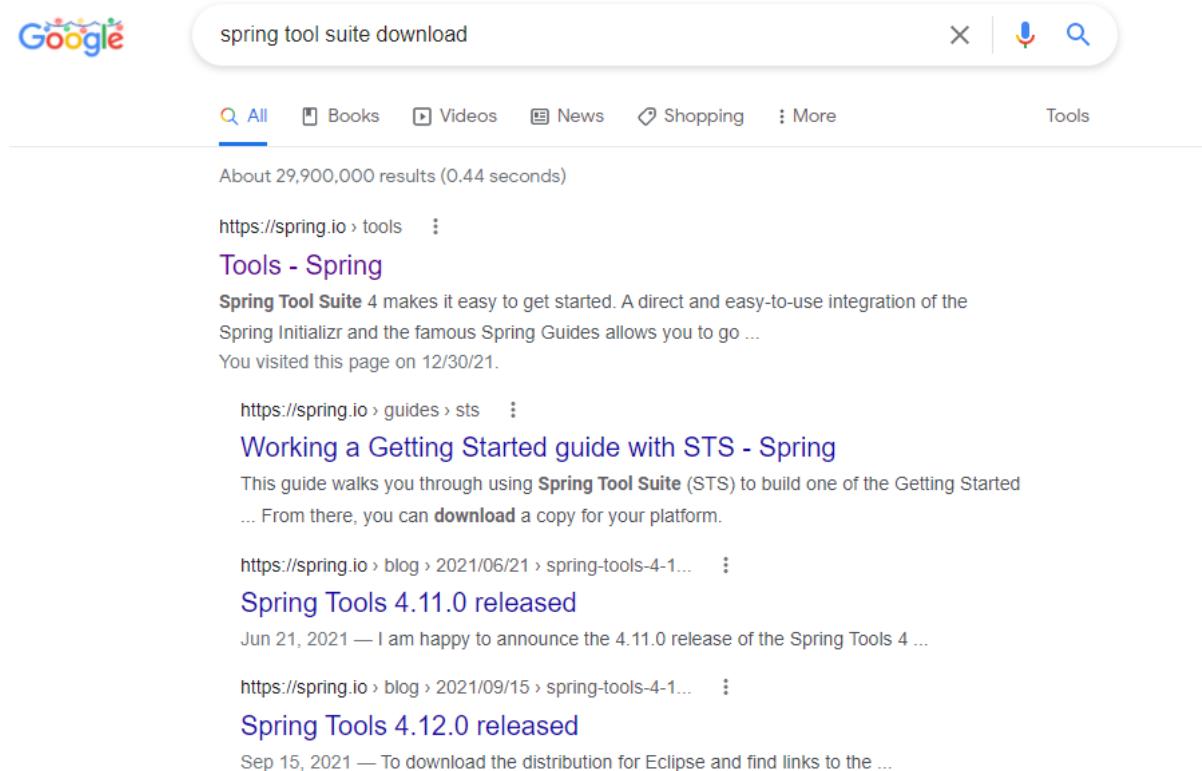
The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

[Linux](#) [macOS](#) [Windows](#)

Product/file description	File size	Download
x64 Compressed Archive	170.66 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256 ↗)
x64 Installer	152 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256 ↗)
x64 MSI Installer	150.89 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256 ↗)

Download Spring Tool Suite

- Click on the first link ‘Tools – Spring’



A screenshot of a Google search results page. The search query "spring tool suite download" is entered in the search bar. The results show approximately 29,900,000 results. The top result is a link to the official Spring Tools website under the heading "Tools - Spring". Below it, another result is titled "Working a Getting Started guide with STS - Spring". Further down, there are news articles about the release of Spring Tools 4.11.0 and 4.12.0.

spring tool suite download

All Books Videos News Shopping More Tools

About 29,900,000 results (0.44 seconds)

<https://spring.io/tools> ::

Tools - Spring

Spring Tool Suite 4 makes it easy to get started. A direct and easy-to-use integration of the Spring Initializr and the famous Spring Guides allows you to go ...

You visited this page on 12/30/21.

<https://spring.io/guides/sts> ::

Working a Getting Started guide with STS - Spring

This guide walks you through using **Spring Tool Suite (STS)** to build one of the Getting Started ... From there, you can **download** a copy for your platform.

<https://spring.io/blog/2021/06/21/spring-tools-4-1...> ::

Spring Tools 4.11.0 released

Jun 21, 2021 — I am happy to announce the 4.11.0 release of the Spring Tools 4 ...

<https://spring.io/blog/2021/09/15/spring-tools-4-1...> ::

Spring Tools 4.12.0 released

Sep 15, 2021 — To download the distribution for Eclipse and find links to the ...

Download Spring Tools 4 for Eclipse for your O/S

- Download the Spring Tools 4 for Eclipse for your OS



Spring Tools 4 for Eclipse

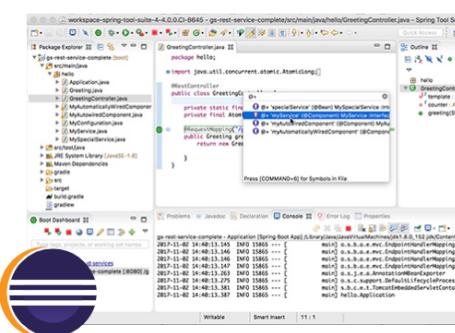
The all-new Spring Tool Suite 4.
Free. Open source.

4.13.0 - LINUX X86_64

4.13.0 - MACOS X86_64

4.13.0 - MACOS ARM_64

4.13.0 - WINDOWS X86_64

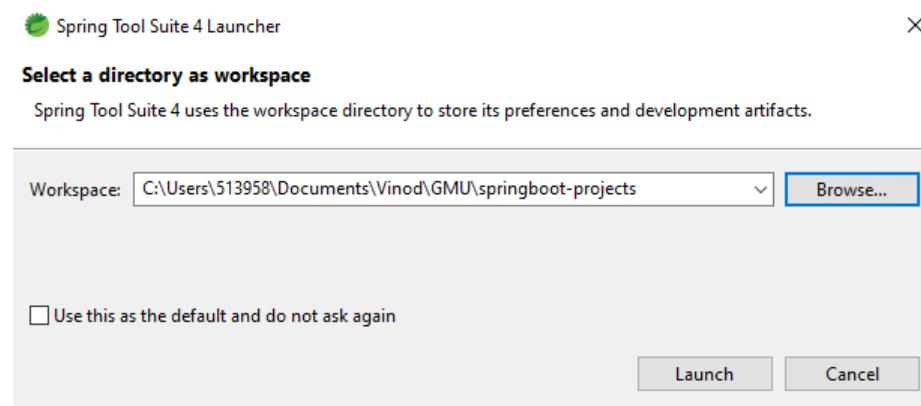
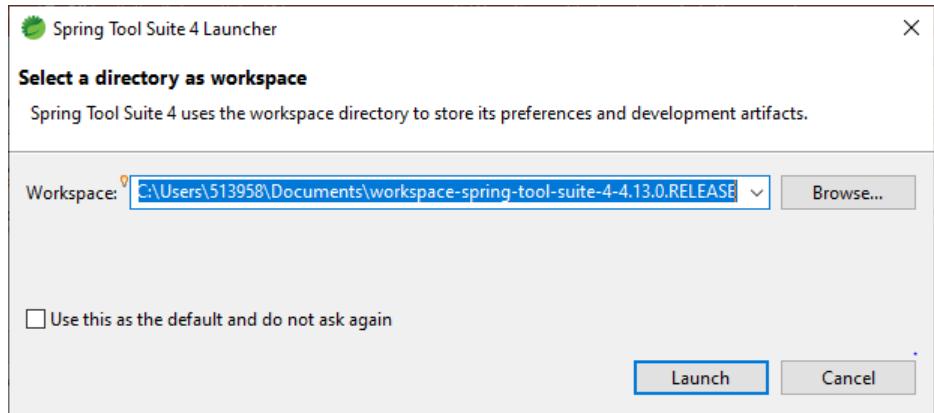


Extract the downloaded Jar

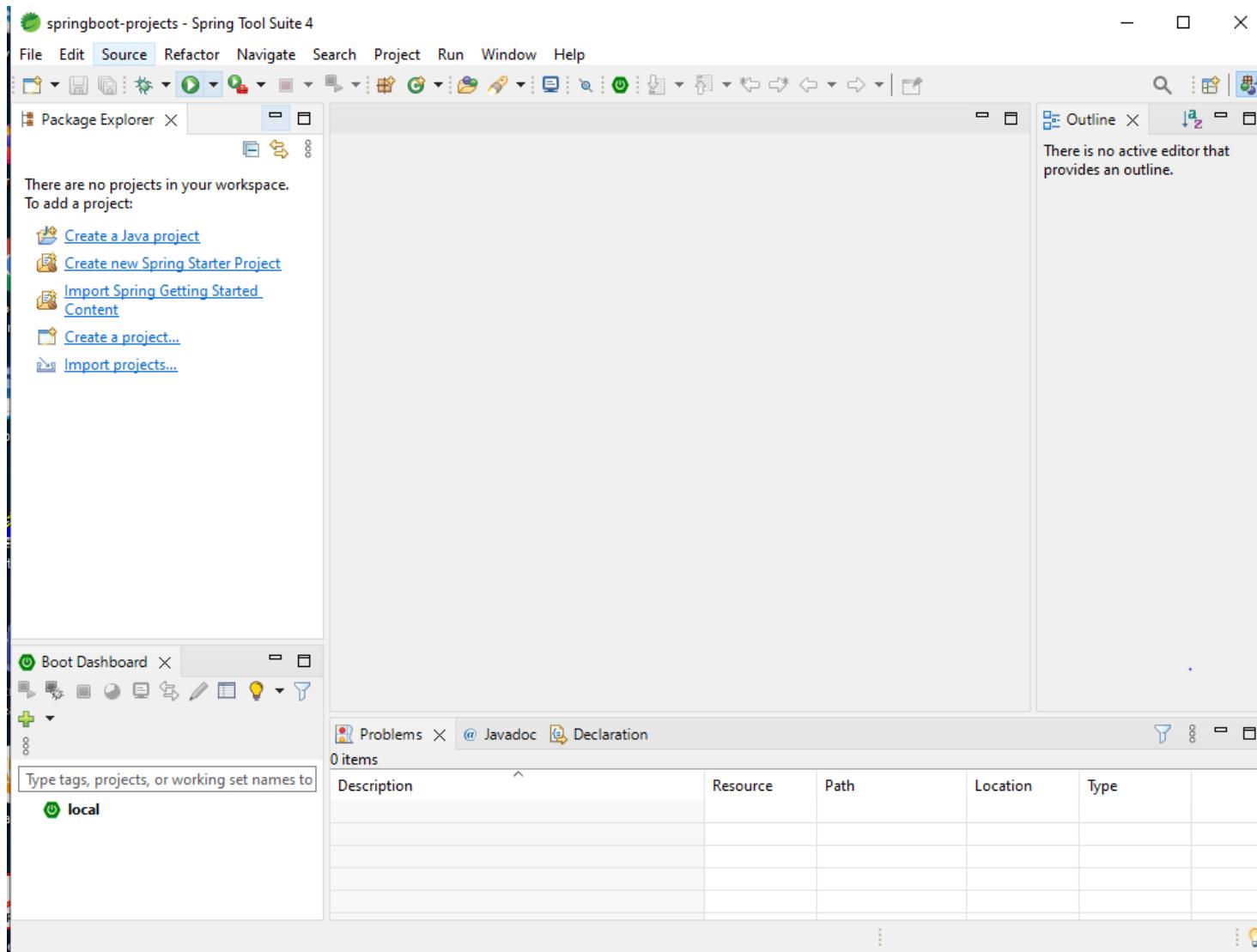
- Double click on **SpringToolSuite4.exe** to open the IDE

ws (C:) > Users > 513958 > Downloads > sts-4.13.0.RELEASE				
	Name	Date modified	Type	Size
	configuration	12/30/2021 12:34 PM	File folder	
	dropins	12/30/2021 12:30 PM	File folder	
	features	12/30/2021 12:31 PM	File folder	
	META-INF	12/30/2021 12:33 PM	File folder	
	p2	12/30/2021 12:30 PM	File folder	
	plugins	12/30/2021 12:33 PM	File folder	
	readme	12/30/2021 12:33 PM	File folder	
	.eclipseproduct	12/30/2021 12:33 PM	ECLIPSEPRODUCT...	1 KB
	artifacts.xml	12/30/2021 12:33 PM	XML Document	158 KB
	eclipsec.exe	12/30/2021 12:33 PM	Application	231 KB
	license.txt	12/30/2021 12:33 PM	Text Document	12 KB
	open-source-licenses.txt	12/30/2021 12:33 PM	Text Document	837 KB
	SpringToolSuite4.exe	12/30/2021 12:33 PM	Application	518 KB
	SpringToolSuite4.ini	12/30/2021 12:31 PM	Configuration sett...	1 KB

Select a directory as workspace and Launch STS IDE

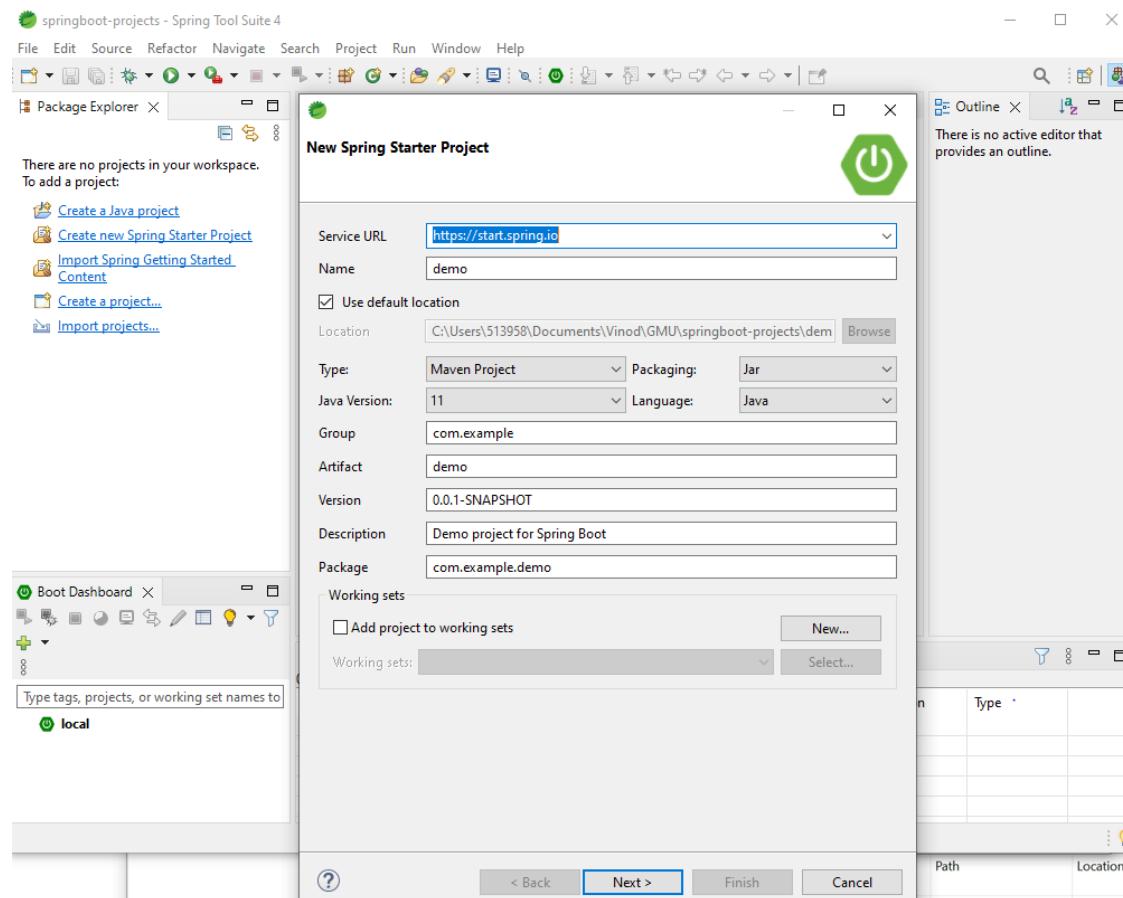


Click on “Create new Spring Starter Project”



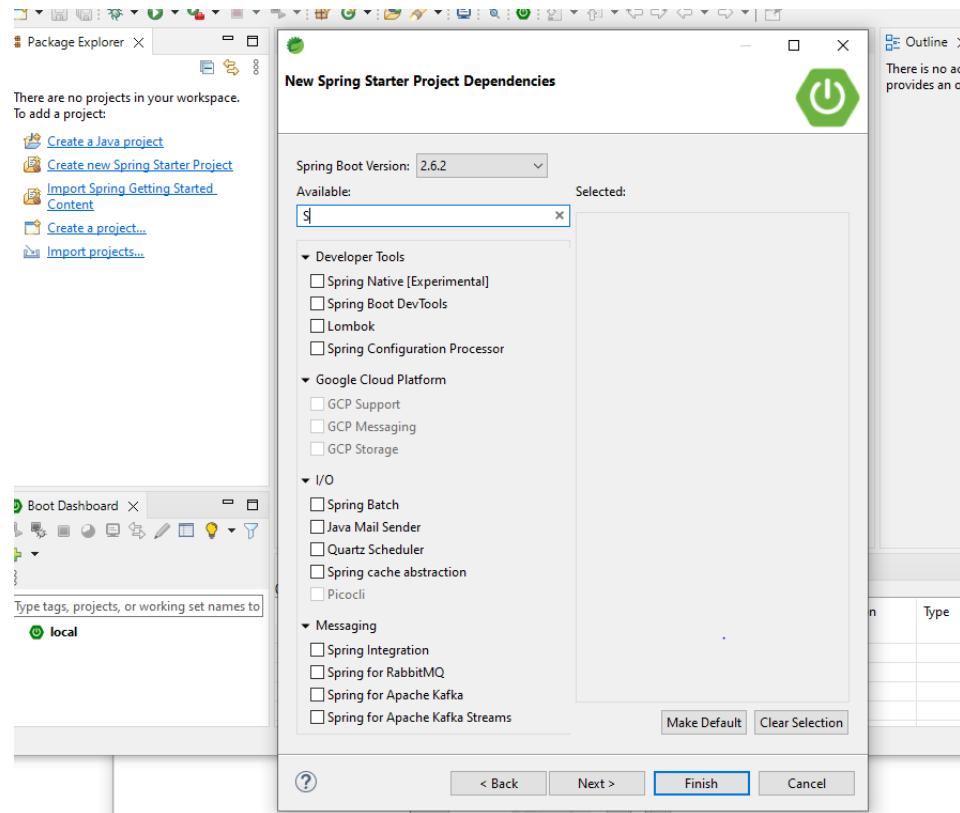
Click on “Create new Spring Starter Project”

- Service URL refers to Spring Initializer – integrated in the STS IDE
- Click on Next and select dependencies



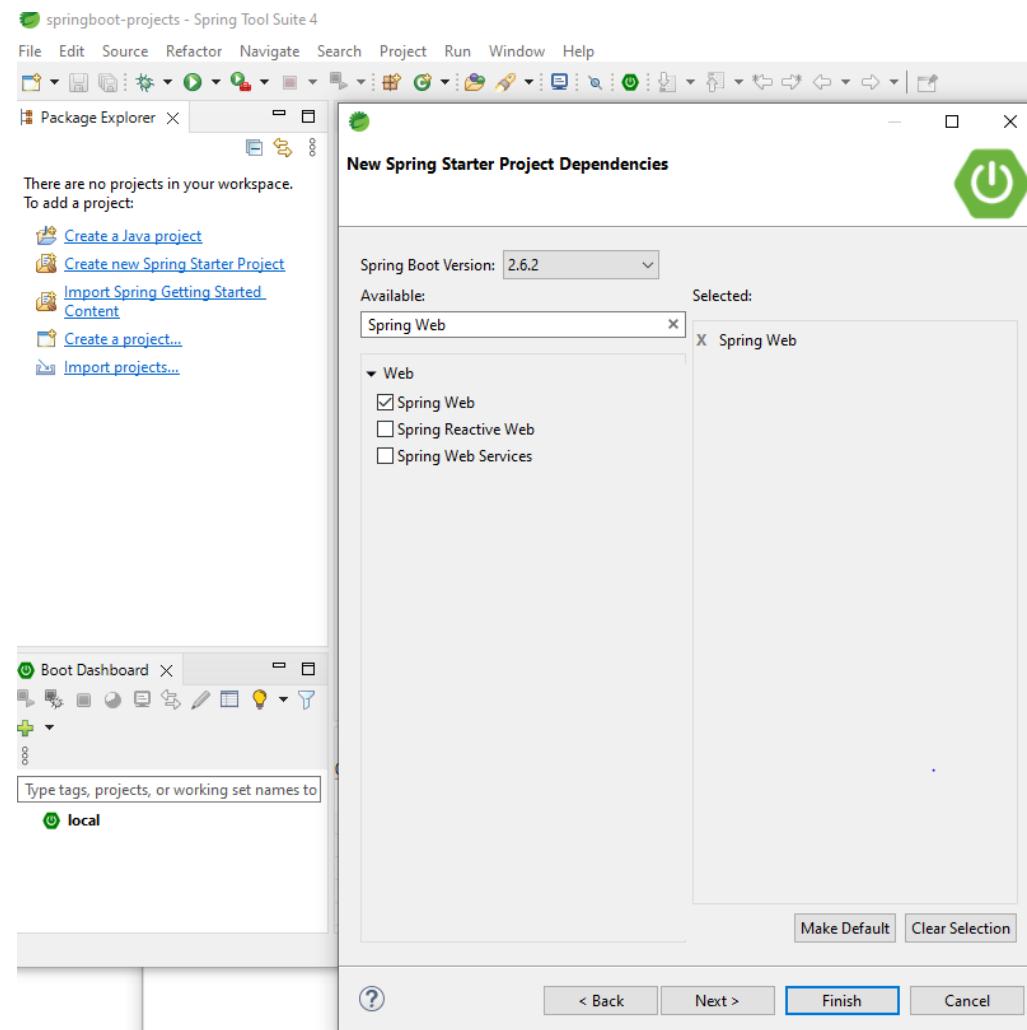
Select Version and project dependencies

- Select latest Spring Boot Version and project dependencies, such as
 - Spring Web, Spring Data JPA, MySQL Driver



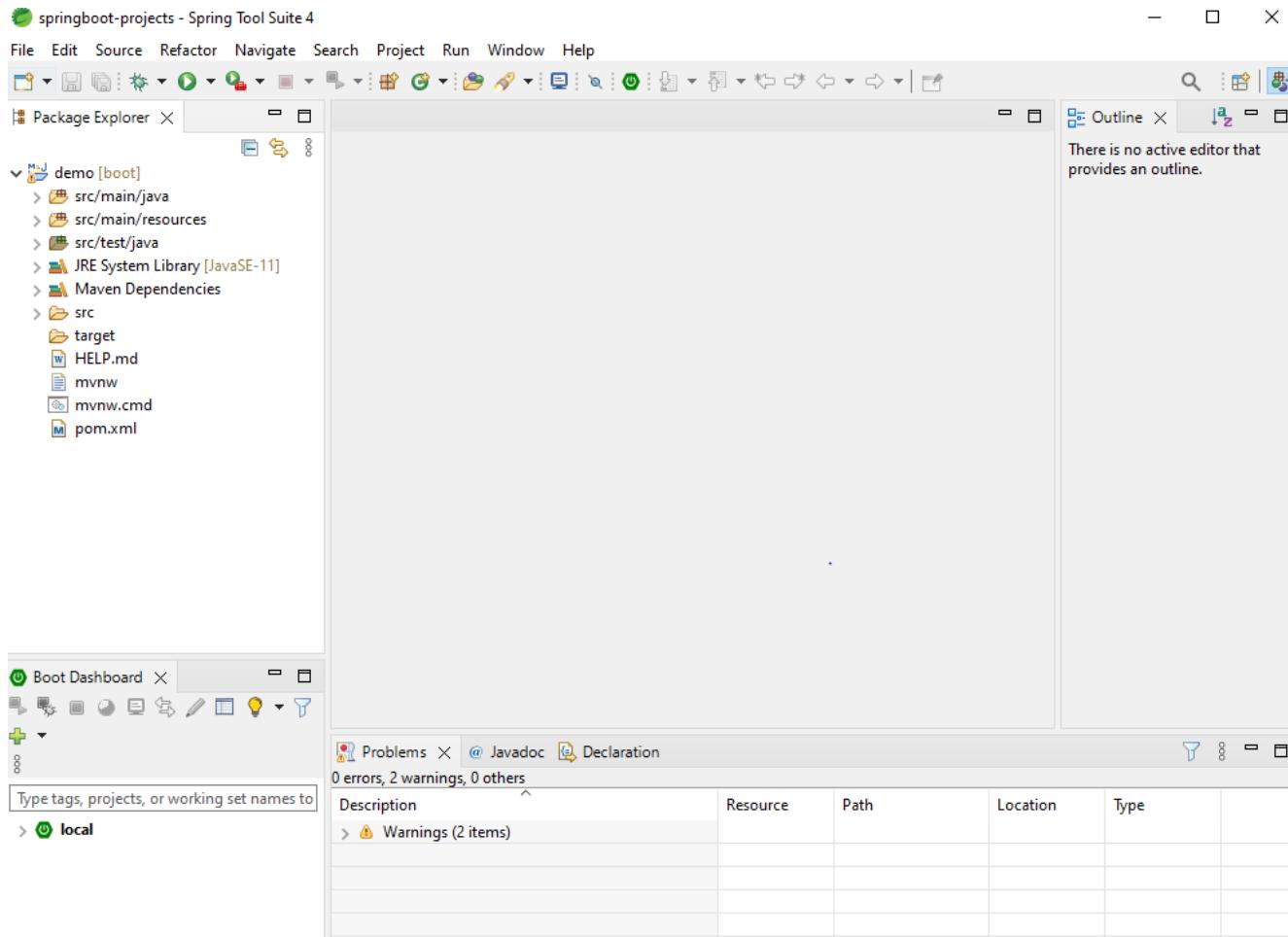
Select Version and project dependencies

- Search and choose ‘Spring Web’ for starter dependencies
- And click Finish



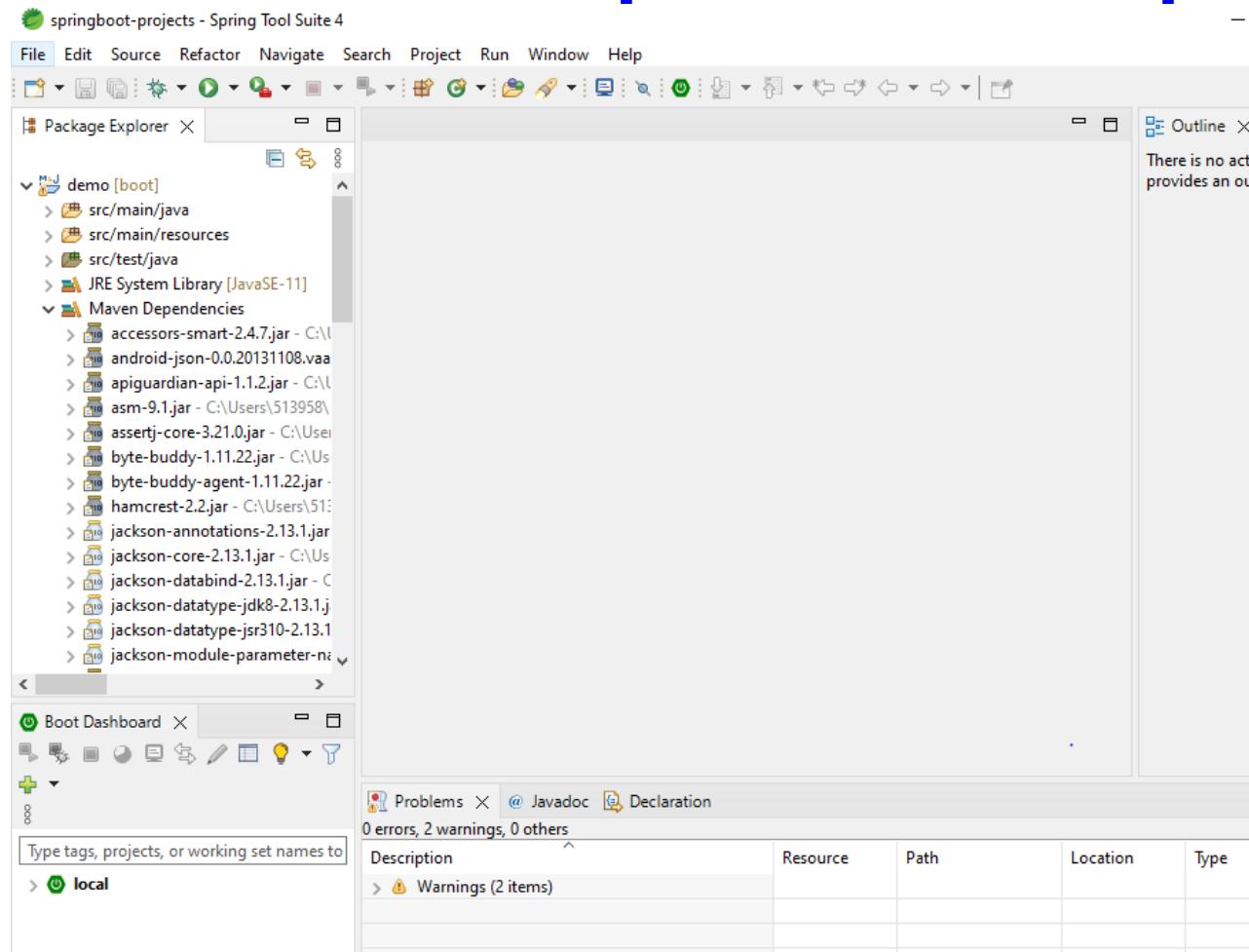
Select Version and project dependencies

- Spring downloads all dependencies and stores in Maven Dependencies repository



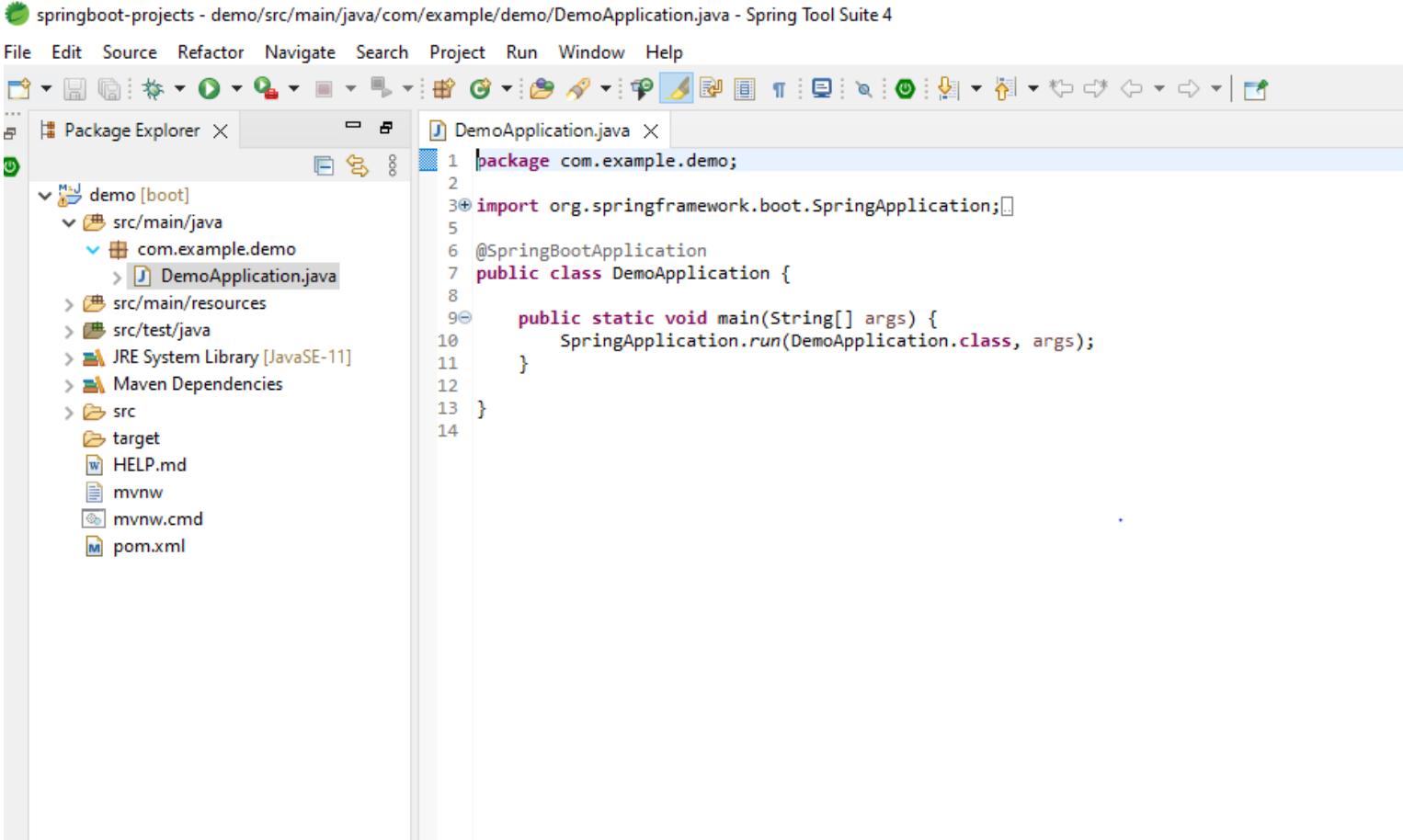
Select Version and project dependencies

- Spring downloads all dependencies and stores in Maven Dependencies repository



DemoApplication.java

- Open **DemoApplication.java** under /src/main/java/com.example.demo
- This is **Spring Boot main class**

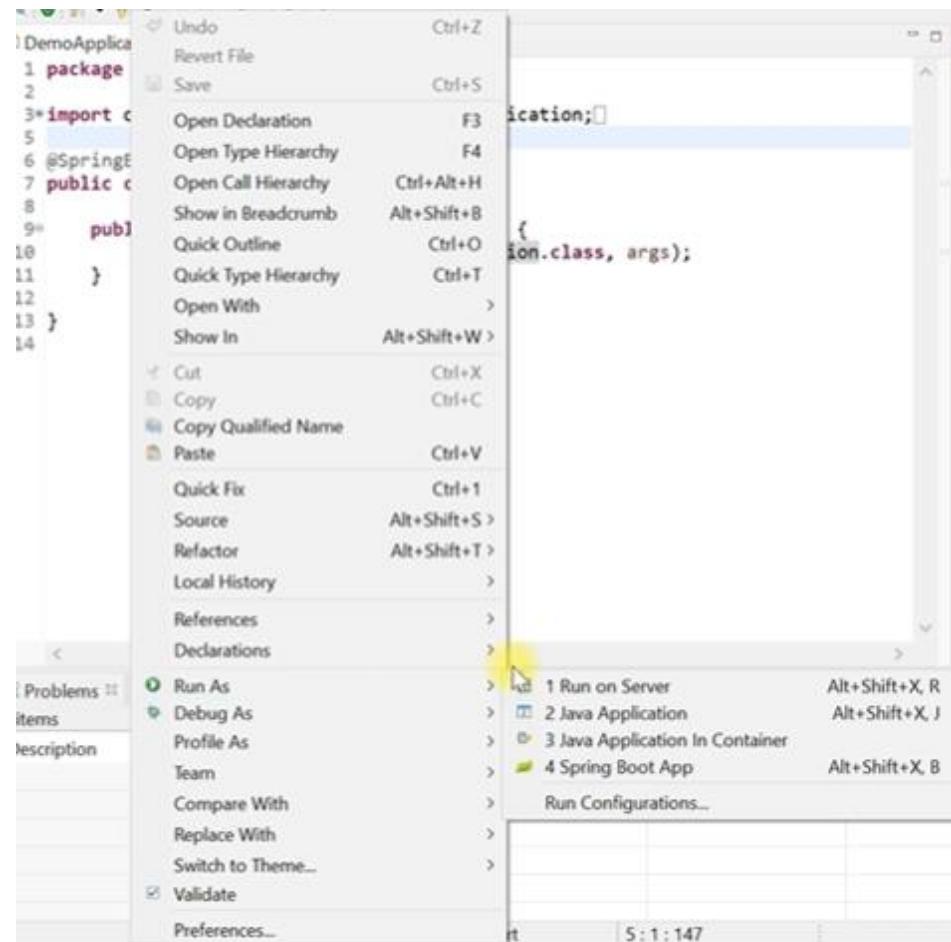


The screenshot shows the Spring Tool Suite 4 interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like New, Open, Save, and Run. The left sidebar is the Package Explorer, showing a project named 'demo [boot]' with a 'src/main/java' folder containing 'com.example.demo' and 'DemoApplication.java'. Other folders like 'src/main/resources', 'src/test/java', and 'src' are also visible. The right side is the code editor for 'DemoApplication.java', displaying the following code:

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
12
13 }
```

Right click in the righthand pane and select Run As Spring Boot Application
Review the console for output including a line saying 'Tomcat started'
This simple Spring Boot app is running on embedded Tomcat at port 8080.

- Right click in the righthand pane and select Run As Spring Boot Application



Right click in the righthand pane and select Run As Spring Boot Application
Review the console for output including a line saying 'Tomcat started'
This simple Spring Boot app is running on embedded Tomcat at port 8080.

- Review the console for output including a line saying 'Tomcat started'
- This simple Spring Boot app is running on embedded Tomcat at port 8080.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "demo [boot]" containing "src/main/java/com/example/demo/DemoApplication.java".
- Code Editor:** Displays the content of `DemoApplication.java`:1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8 public static void main(String[] args) {
9 SpringApplication.run(DemoApplication.class, args);
10 }
11
12 }
13
14 }
- Console:** Shows the terminal output for the application's startup. It includes a decorative ASCII logo and the following log entries:

```
2021-12-30 14:24:26.030 INFO 42388 --- [           ] : Starting DemoApplication using Java 17.0.1 on CHAN-322
2021-12-30 14:24:26.032 INFO 42388 --- [           ] : No active profile set, falling back to default profile
2021-12-30 14:24:26.908 INFO 42388 --- [           ] : Tomcat initialized with port(s): 8080 (http)
2021-12-30 14:24:26.919 INFO 42388 --- [           ] : Starting service [Tomcat]
2021-12-30 14:24:26.926 INFO 42388 --- [           ] : Starting Servlet engine: [Apache Tomcat/9.0.56]
2021-12-30 14:24:27.039 INFO 42388 --- [           ] : Initializing Spring embedded WebApplicationContext
2021-12-30 14:24:27.039 INFO 42388 --- [           ] : Root WebApplicationContext: initialization completed
2021-12-30 14:24:27.577 INFO 42388 --- [           ] : Tomcat started on port(s): 8080 (http) with context pa
2021-12-30 14:24:27.584 INFO 42388 --- [           ] : Started DemoApplication in 1.894 seconds (JVM running
```

Spring Initializer (<https://start.spring.io>)

- You can use Spring Initializer (<https://start.spring.io>) to create Spring Boot maven project, download and then import maven project in STS IDE
- Please note - *Spring Initializer is now integrated with STS IDE*, so you can create Spring Boot maven project directly inside STS IDE instead of importing it from outside.

The screenshot shows the Spring Initializer web application interface. At the top, there's a navigation bar with icons for GitHub and Twitter, followed by the "spring initializr" logo. Below the navigation, there are three main sections: Project, Language, and Dependencies.

- Project:** Maven Project (selected), Gradle Project.
- Language:** Java (selected), Kotlin, Groovy.
- Dependencies:** A button labeled "ADD DEPENDENCIES... CTRL + B". Below it, a message says "No dependency selected".

Under "Project Metadata", the following fields are filled:

- Group: com.example
- Artifact: demo
- Name: demo
- Description: Demo project for Spring Boot
- Package name: com.example.demo
- Packaging: Jar (selected), War.
- Java version: 17 (selected), 11, 8.

At the bottom, there are buttons for "GENERATE" (CTRL + ⌘) and "EXPLORE" (CTRL + SPACE), and a "SHARE..." button.

Spring Initializer (<https://start.spring.io>)

The current version of Spring Boot changes regularly. Just choose the latest release (but not snapshot).

Click on 'Add dependencies', type 'Web' in the search box, then click on the dependency 'Spring Web' to select it.

Project

- Maven Project
- Gradle Project

Language

- Java
- Kotlin
- Groovy

Dependencies

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Project Metadata

Group: com.example

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package name: com.example.demo

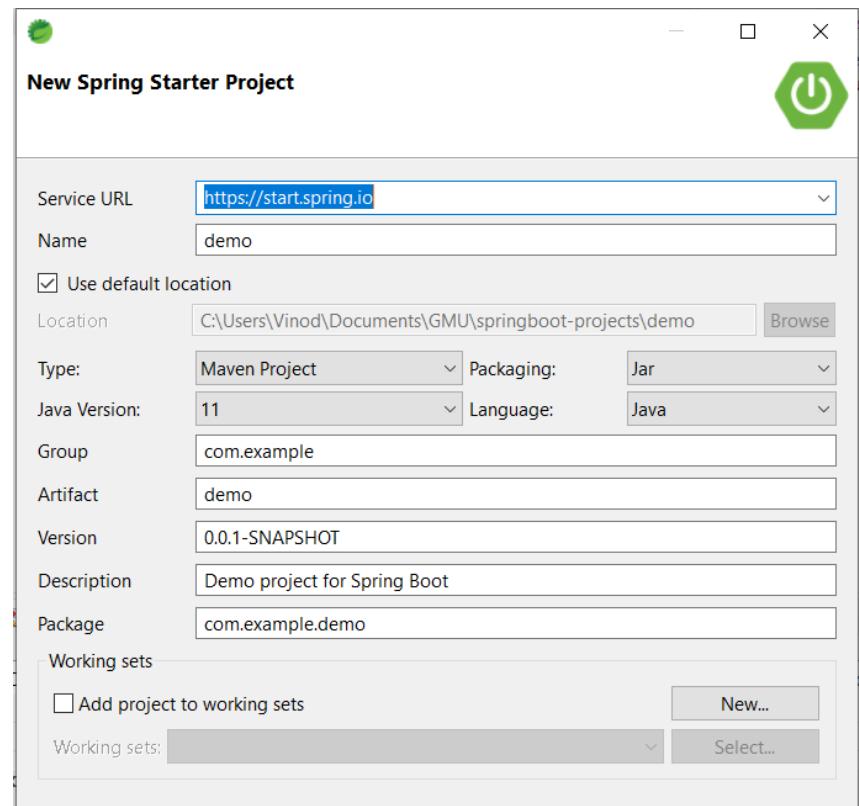
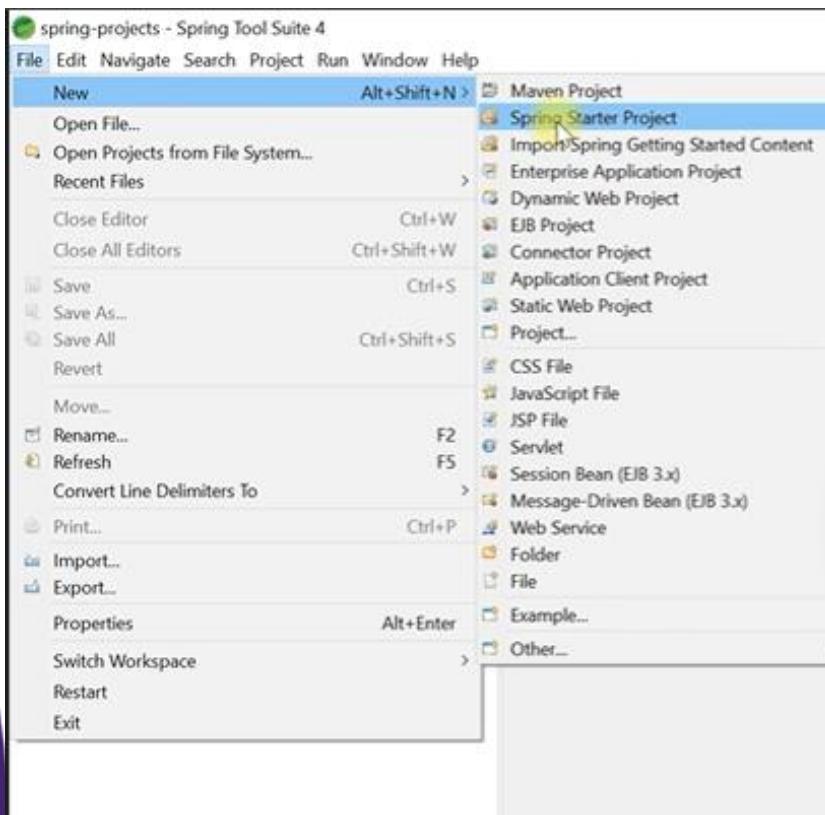
Packaging: Jar War

Java: 14 11 8

GENERATE ⌘ + ↵ EXPLORE CTRL + SPACE SHARE...

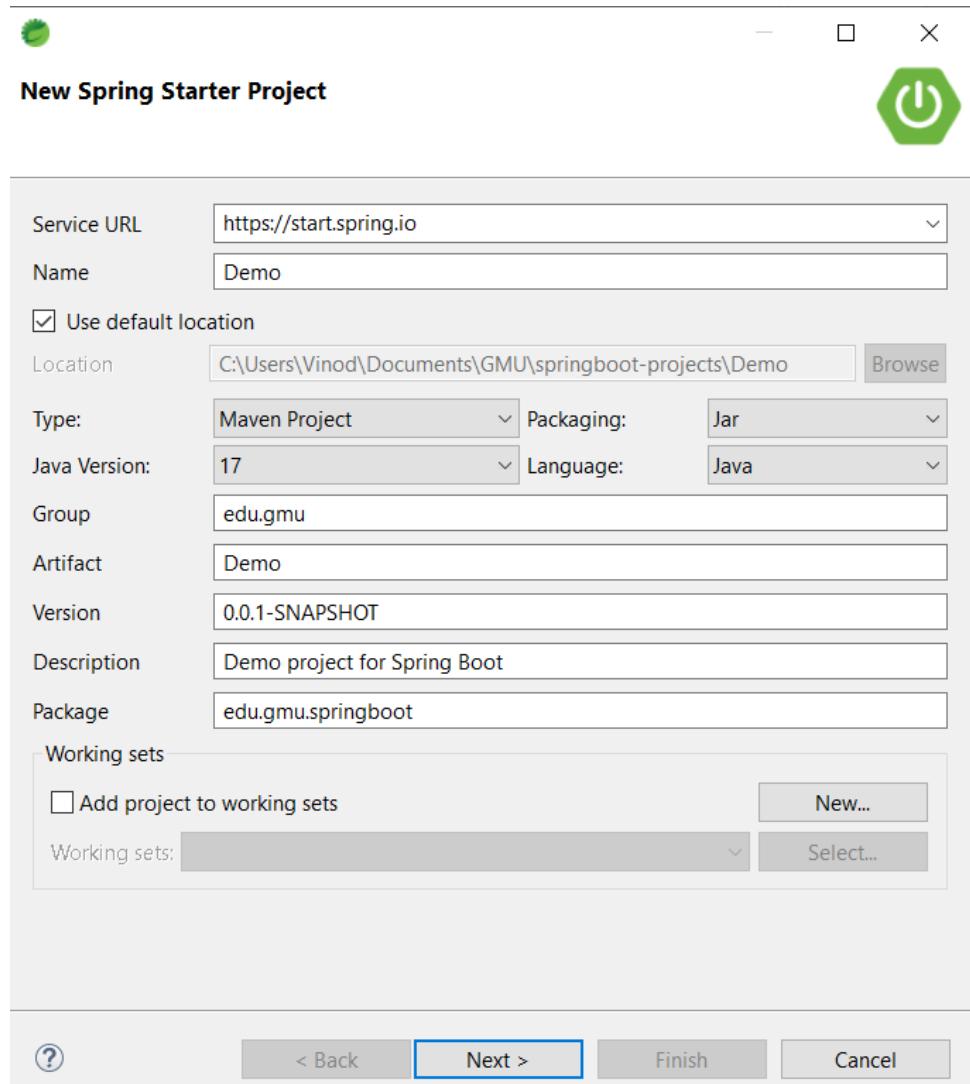
Creating Spring Boot Maven project using STS IDE

- From inside STS IDE, select:
- File -> New -> Spring Starter Project***



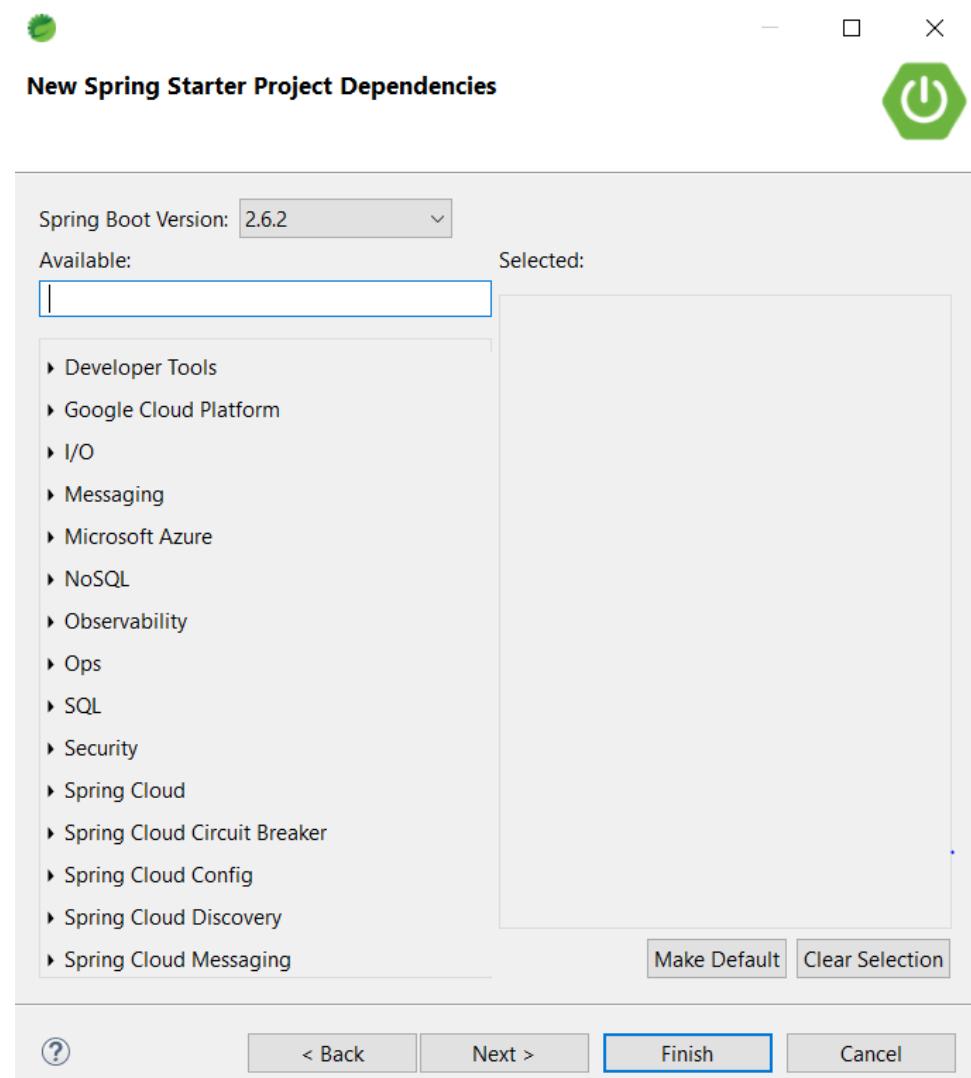
Creating Spring Boot Maven project using STS IDE

- **Select/update values for**
 - Name,
 - type,
 - Java Version,
 - packaging, language,
 - Group,
 - Description,
 - Package
- **Artifact is same as Project Name**
- **Press Next**



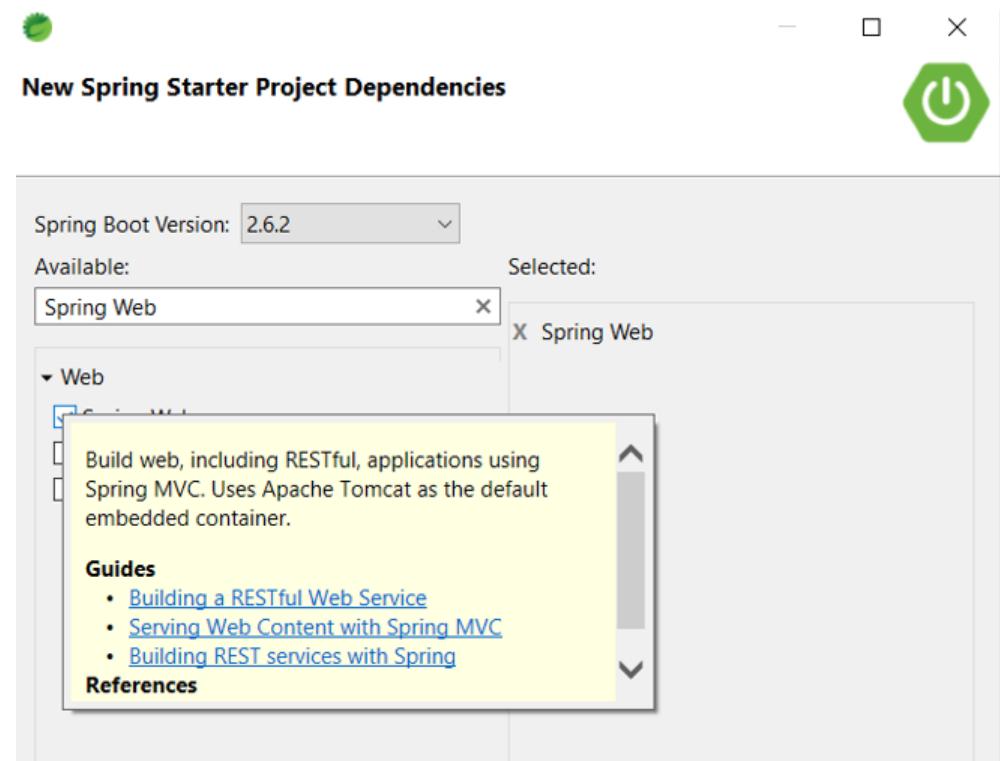
Select project dependencies

- **Select Spring Starter Project Dependencies**
- **For example,**
 - Select search and select **Spring Web** to create web application or **RESTful web service**.
- **And press Next**



Select project dependencies

- **Select Spring Starter Project Dependencies**
- **For example,**
 - Select search and select **Spring Web** to create web application or **RESTful web service**.
- **And press Next**



Select project dependencies

- ***Spring Web***

- Build web, including RESTful applications using Spring MVC.
- Uses Apache Tomcat as the default embedded container.

- **Guides**

- Building a RESTful Web Service
- Serving Web Content with Spring MVC
- Building REST services with Spring

- **References**

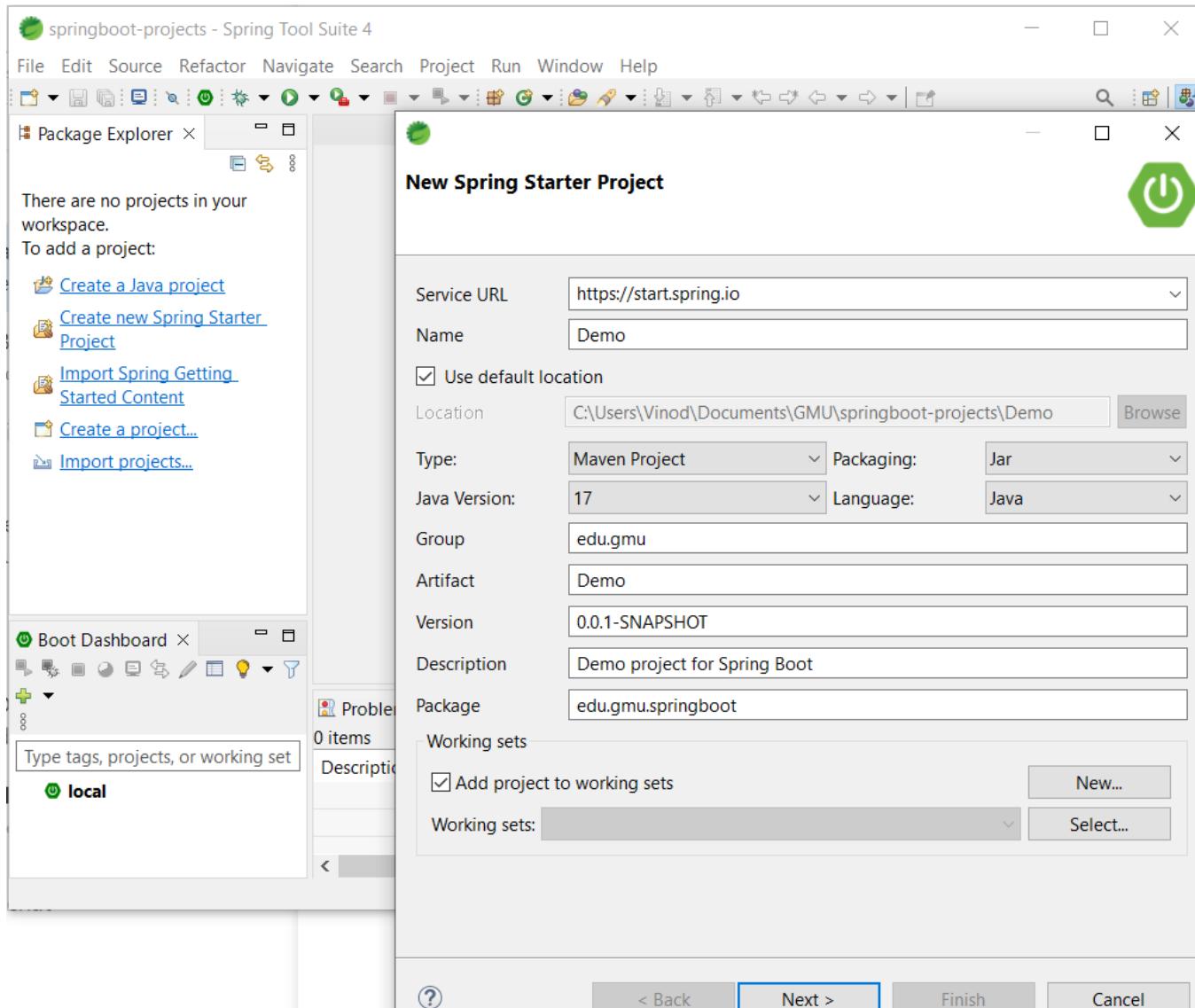
- Spring Boot Reference Doc

Select project dependencies

- ***Spring Data JPA***
 - is an abstraction on top of JPA and internally uses Hibernate as a JPA provider
 - Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- **Guides**
 - Accessing Data with JPA
- **References**
 - Spring Boot Reference Doc
 - Accessing data with MySQL

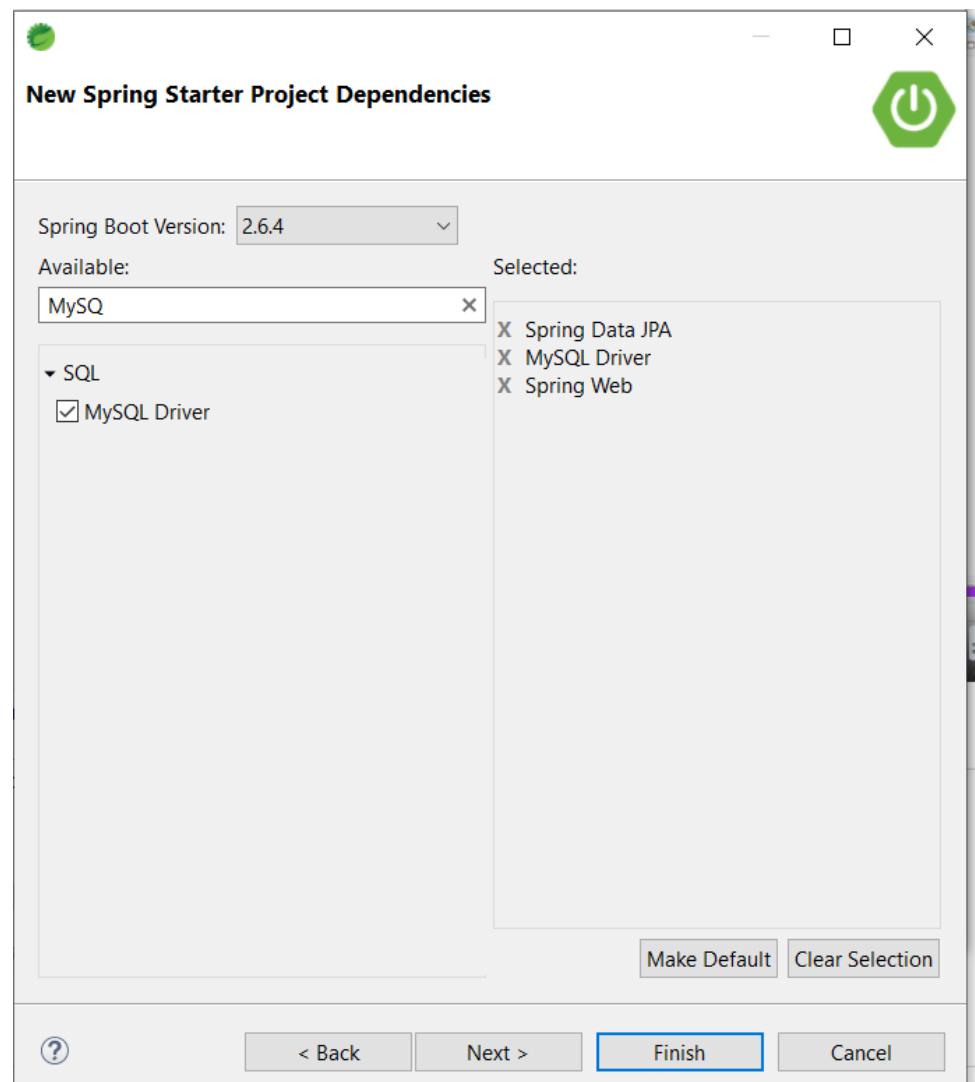
Let's create a Spring Starter Project

- Select File->New -> Spring Starter Project,
- Select values for Name, Group, Select search and select Spring, Artifact, package, java version, and then And press Next



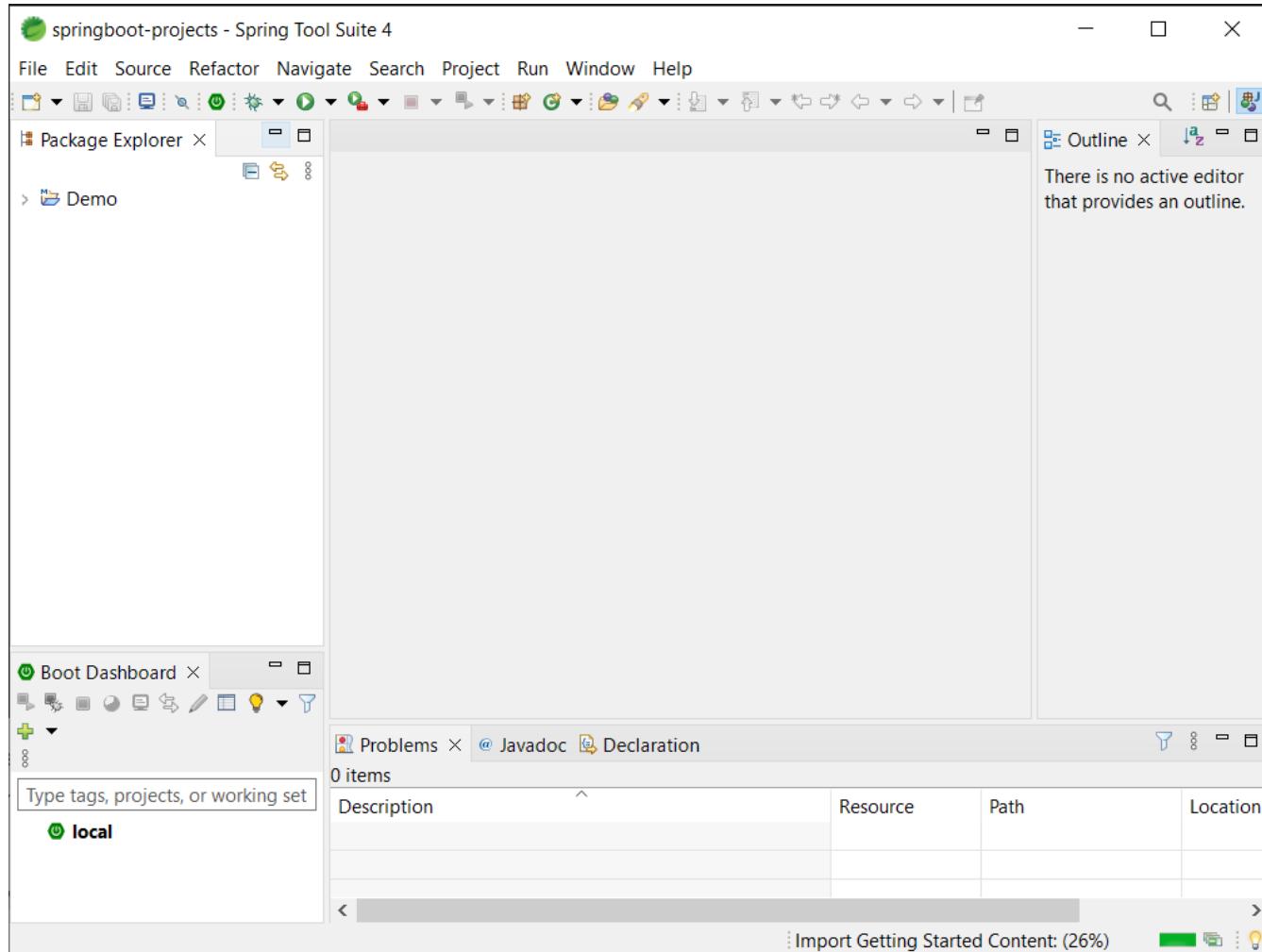
Create Spring Starter Project

- *Select Spring Starter Project Dependencies*
– e.g., In the Available text box, search and select *Spring Web* to create web application or RESTful web service.
- Also, select *Spring Data JPA* as well as *MySQL Driver*
- And press *Finish*



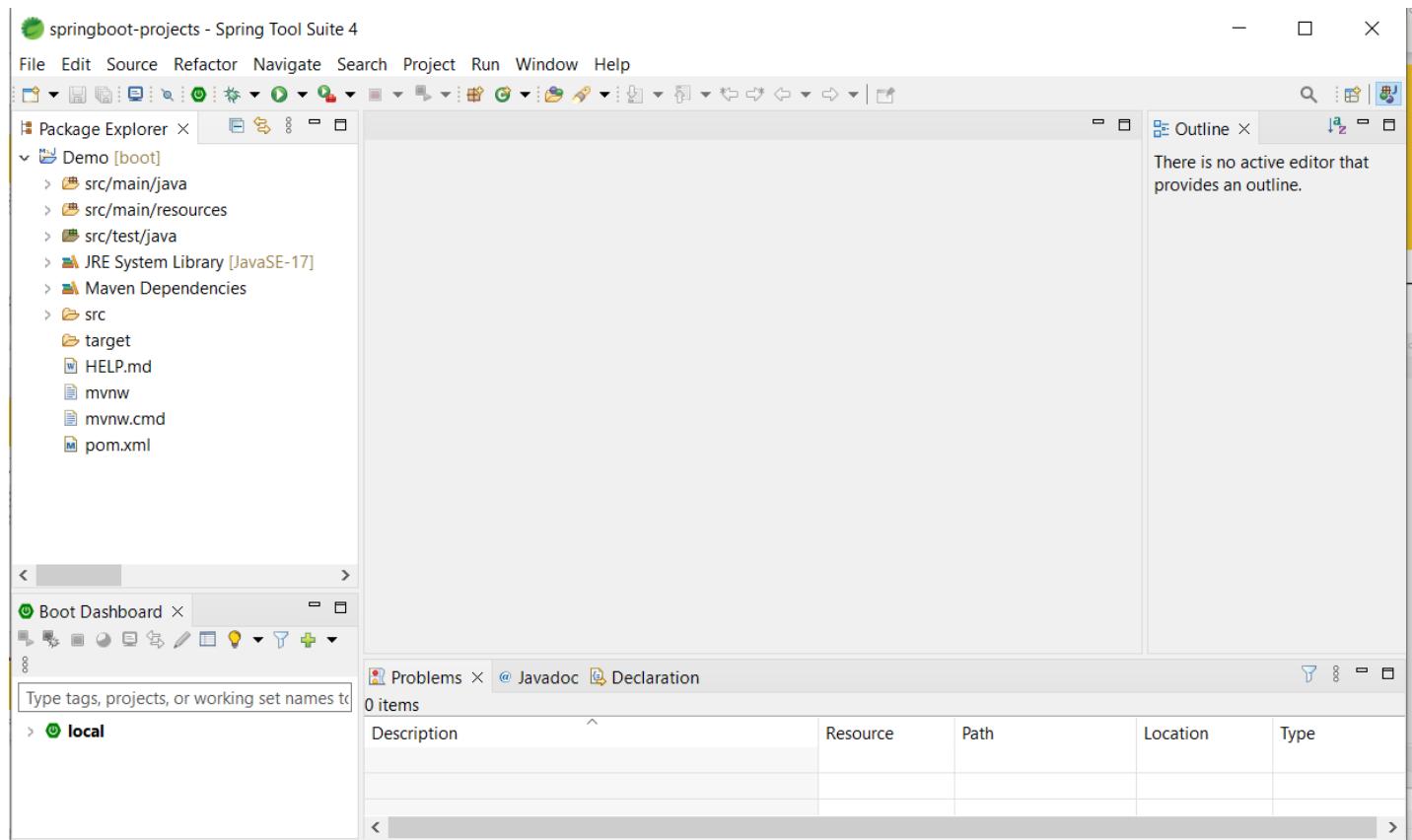
Create Spring Starter Project

*Maven downloads
all dependencies
from the Internet*

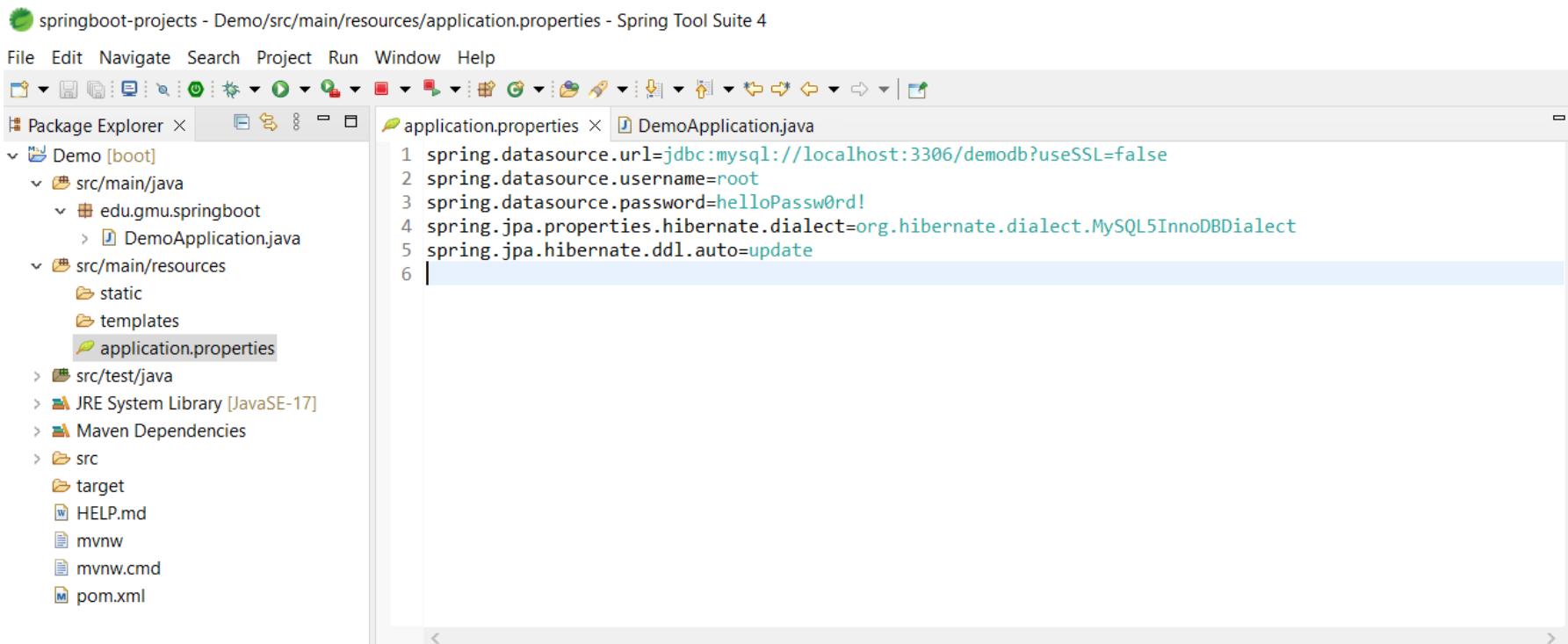


Create Spring Starter Project

*Maven
downloads all
dependencies
from the
Internet and
stores them
under [Maven
Dependencies
folder](#)*



Update application.properties with database/jpa properties



The screenshot shows the Spring Tool Suite 4 interface. The title bar reads "springboot-projects - Demo/src/main/resources/application.properties - Spring Tool Suite 4". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar is the "Package Explorer" showing the project structure:

- Demo [boot]
 - src/main/java
 - edu.gmu.springboot
 - DemoApplication.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - src/test/java
 - JRE System Library [JavaSE-17]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

Run the application

DemoApplication->Run As -> Spring Boot Application

The Console shows the Spring Boot application is running on Tomcat and connected to MySQL database using JPA/Hibernate.

The screenshot shows the Spring Tool Suite 4 interface with the following details:

- Project Structure:** The Package Explorer shows the project structure for "Demo [boot]". It includes the `src/main/java` directory containing `DemoApplication.java` and `application.properties`, and the `src/main/resources` directory containing `static`, `templates`, and `application.properties` files.
- Code Editor:** The main editor shows the `DemoApplication.java` code:

```
1 package edu.gmu.springboot;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
12
13 }
```
- Outline View:** The Outline view on the right shows the class structure: `edu.gmu.springboot` > `DemoApplication` > `main(String[]) : void`.
- Eclipse Console:** The bottom pane displays the console output for the "Demo - DemoApplication [Spring Boot App]" session. It shows the Spring Boot logo and the command-line arguments used to run the application.
- Log Output:** The console log shows the application's startup process, including the bootstrapping of Spring Data JPA repositories, the initialization of Tomcat, and the creation of the HikariDataSource.

```
2022-03-15 23:13:39.868 INFO 14400 --- [           main] edu.gmu.springboot.DemoApplication      : Starting DemoApplication using Java 17.0.1 on DESI
2022-03-15 23:13:39.872 INFO 14400 --- [           main] edu.gmu.springboot.DemoApplication      : No active profile set, falling back to 1 default: 
2022-03-15 23:13:40.802 INFO 14400 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEF
2022-03-15 23:13:40.830 INFO 14400 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 10 ms
2022-03-15 23:13:41.453 INFO 14400 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-03-15 23:13:41.471 INFO 14400 --- [           main] o.apache.catalina.core.StandardService  : Starting service [Tomcat]
2022-03-15 23:13:41.471 INFO 14400 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.58]
2022-03-15 23:13:41.637 INFO 14400 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-03-15 23:13:41.638 INFO 14400 --- [           main] w.s.c.ServletWebServerApplicationContext  : Root WebApplicationContext: initialization complete
2022-03-15 23:13:41.832 INFO 14400 --- [           main] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Starting...
```

Run the application

DemoApplication->Run As -> Spring Boot Application

The Console shows the Spring Boot application is running on Tomcat and connected to MySQL database using JPA/Hibernate.

The screenshot shows the Eclipse IDE interface for Spring Tool Suite 4. The left side features the Package Explorer with a project named 'Demo [boot]' containing packages like 'edu.gmu.springboot' and files like 'DemoApplication.java'. The central area displays the code for 'DemoApplication.java':

```
1 package edu.gmu.springboot;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
```

A blue oval highlights the line 'SpringApplication.run(DemoApplication.class, args);'. Handwritten blue text above the code says 'R-click → Run as Spring Boot'. A blue arrow points from the handwritten text to the right edge of the code editor. Another blue arrow points from the handwritten text to the 'Run' button in the toolbar at the bottom right of the interface. The right side of the interface includes the Outline view and the Eclipse Console window, which shows the application's log output:

```
2022-03-27 14:16:59.955 INFO 31740 --- [           main] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Starting
2022-03-27 14:17:00.123 INFO 31740 --- [           main] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Start completed.
2022-03-27 14:17:00.172 INFO 31740 --- [           main] o.hibernat.jpa.internal.util.LogHelper   : HHH000204: Processing PersistenceUnitInfo
2022-03-27 14:17:00.240 INFO 31740 --- [           main] org.hibernate.Version                   : HHH000412: Hibernate Core version 5.6
2022-03-27 14:17:00.425 INFO 31740 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations version 5.1
2022-03-27 14:17:00.547 INFO 31740 --- [           main] org.hibernate.dialect.Dialect          : HHH000400: Using dialect org.hibernate.dialect
2022-03-27 14:17:00.731 INFO 31740 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation
2022-03-27 14:17:00.740 INFO 31740 --- [           main] j.LocalContainerEntityManagerFactoryBean  : Initialized JPA EntityManagerFactory for persistence unit 'Default'
2022-03-27 14:17:00.772 WARN 31740 --- [           main] JpaBaseConfiguration$JpaWebConfiguration  : spring.jpa.open-in-view is enabled by default, so database queries may be performed during view rendering
2022-03-27 14:17:01.115 INFO 31740 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-03-27 14:17:01.122 INFO 31740 --- [           main] edu.gmu.springboot.DemoApplication       : Started DemoApplication in 2.978 seconds
```

Create JPA Student Entity and Repository

*Create JPA Entity called Student under package edu.gmu.springboot.model
(Right click on package name and Create a java class and add annotations)*

The screenshot shows the Spring Tool Suite 4 interface. On the left, the Package Explorer view displays the project structure under the 'Demo [boot]' package. The 'src/main/java' folder contains 'edu.gmu.springboot' and 'edu.gmu.springboot.model'. The 'edu.gmu.springboot.model' folder has a file named 'Student.java' which is currently selected. On the right, the code editor shows the 'DemoApplication.java' file and the 'Student.java' file. The 'Student.java' code is as follows:

```
1 package edu.gmu.springboot.model;
2
3 import java.util.Objects;
4
5 import javax.persistence.Column;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.Table;
11
12 @Entity
13 @Table(name="students")
14 public class Student {
15     @Id
16     @GeneratedValue(strategy=GenerationType.IDENTITY)
17     private long id;
18
19     @Column(name="first_name", nullable=false)
20     private String firstName;
21
22     @Column(name="last_name", nullable=false)
23     private String lastName;
24 }
```

The code editor's status bar at the bottom says 'Type tags, projects, or working set names to match (inc...)'.

Create JPA Student Entity and Repository (contd.)

*Create JPA Entity called Student under package edu.gmu.springboot.model
(Right click on package name and Create a java class and add annotations)*

```
25@Column(name="email", nullable=false)
26    private String email;
27
28    public long getId() {
29        return id;
30    }
31    public void setId(long id) {
32        this.id = id;
33    }
34    public String getFirstName() {
35        return firstName;
36    }
37    public void setFirstName(String firstName) {
38        this.firstName = firstName;
39    }
40    public String getLastName() {
41        return lastName;
42    }
43    public void setLastName(String lastName) {
44        this.lastName = lastName;
45    }
```

Create JPA Student Entity and Repository (contd)

*Create JPA Entity called Student under package edu.gmu.springboot.model
(Right click on package name and Create a java class and add annotations)*

```
49@  public void setEmail(String email) {  
50    this.email = email;  
51}  
52  
53@ Override  
54 public String toString() {  
55    return "Student [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", emai  
56}  
57@ Override  
58 public int hashCode() {  
59    return Objects.hash(email, firstName, id, lastName);  
60}  
61@ Override  
62 public boolean equals(Object obj) {  
63    if (this == obj)  
64        return true;  
65    if (obj == null)  
66        return false;  
67    if (getClass() != obj.getClass())  
68        return false;  
69    Student other = (Student) obj;  
70    return Objects.equals(email, other.email) && Objects.equals(firstName, other.firstName) && i  
71        && Objects.equals(lastName, other.lastName);  
72}
```

Create JPA Student Entity and Repository

If you *run the application as Spring Boot*, highlighting and right-clicking on *DemoApplication* and selecting *run as Spring Boot*. Should be resulted in creating *students database*

The screenshot shows the Spring Tool Suite 4 interface with the following details:

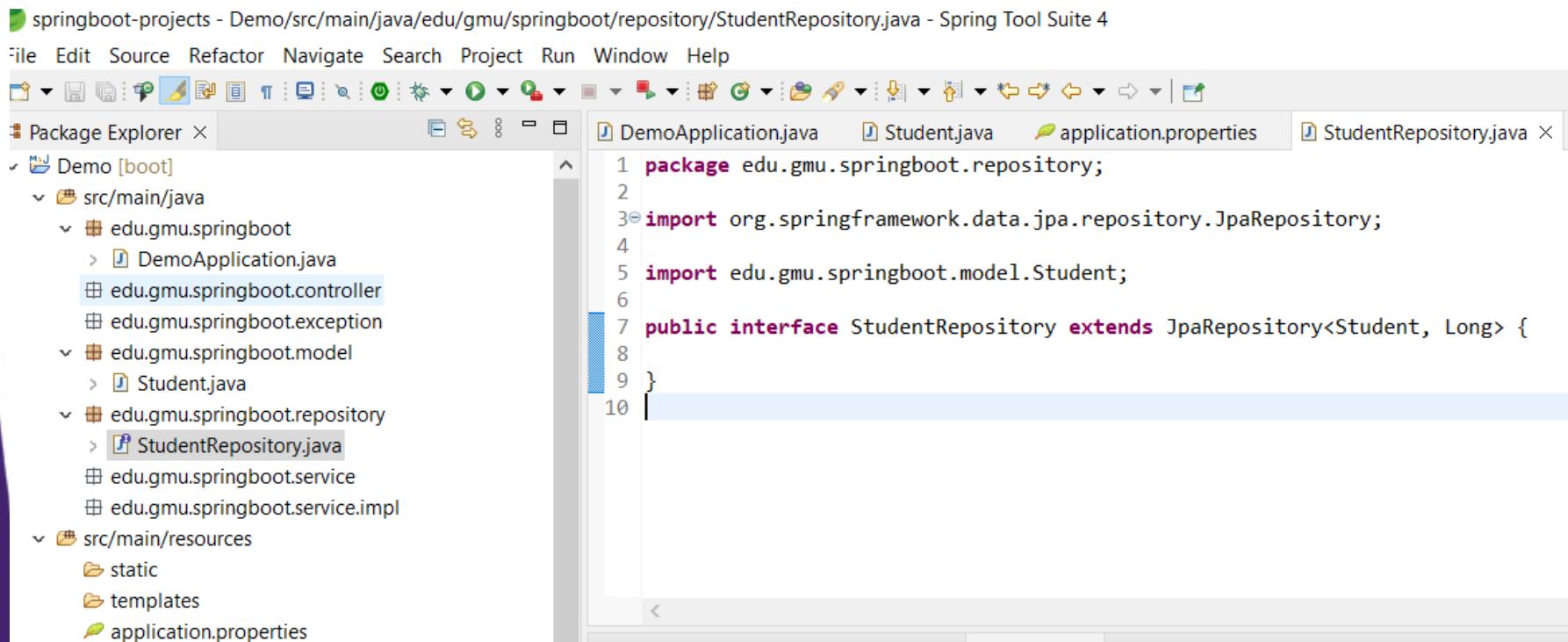
- File Explorer:** Shows the project structure under the package `Demo [boot]`. It includes `src/main/java` with `edu.gmu.springboot` containing `DemoApplication.java`, `controller`, `exception`, `model` (containing `Student.java`), `repository`, `service`, and `service.impl`; and `src/main/resources` with `static`, `templates`, `application.properties`, and `src/test/java`.
- Code Editor:** Displays the `DemoApplication.java` file with the following code:

```
1 package edu.gmu.springboot;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
```
- Outline View:** Shows the `DemoApplication` class with its `main(String[])` method.
- Console:** Shows the output of running the application:

```
Demo - DemoApplication [Spring Boot App]
Starting Servlet engine: [Apache Tomcat/9.0.58]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1072 ms
HikariPool-1 - Starting...
HikariPool-1 - Start completed.
HHH000204: Processing PersistenceUnitInfo [name: default]
HHH000412: Hibernate ORM core version 5.6.5.Final
HCANN00001: Hibernate Commons Annotations {5.1.2.Final}
HHH000400: Using dialect: org.hibernate.dialect.MySQLInnoDBDialect
HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Initialized JPA EntityManagerFactory for persistence unit 'default'
spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring
Tomcat started on port(s): 8080 (http) with context path ''
Started DemoApplication in 2.923 seconds (JVM running for 3.597)
```

Create Student Repository Interface

- Create `StudentRepository` interface under the repository package by *extending JpaRepository*,
- *JpaRepository requires two parameters – type of entity, `Student`, and the second parameter is type of primary key, `long` in this example.*
- Now `StudentRepository` will have *CRUD methods* for student.
 - *The Student JPA repository allows to implement and perform CRUD operation on Student jpa entity*



The screenshot shows the Spring Tool Suite 4 interface. The top bar displays the title "springboot-projects - Demo/src/main/java/edu/gmu/springboot/repository/StudentRepository.java - Spring Tool Suite 4". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, and Run. The left side features the "Package Explorer" view, which shows the project structure:

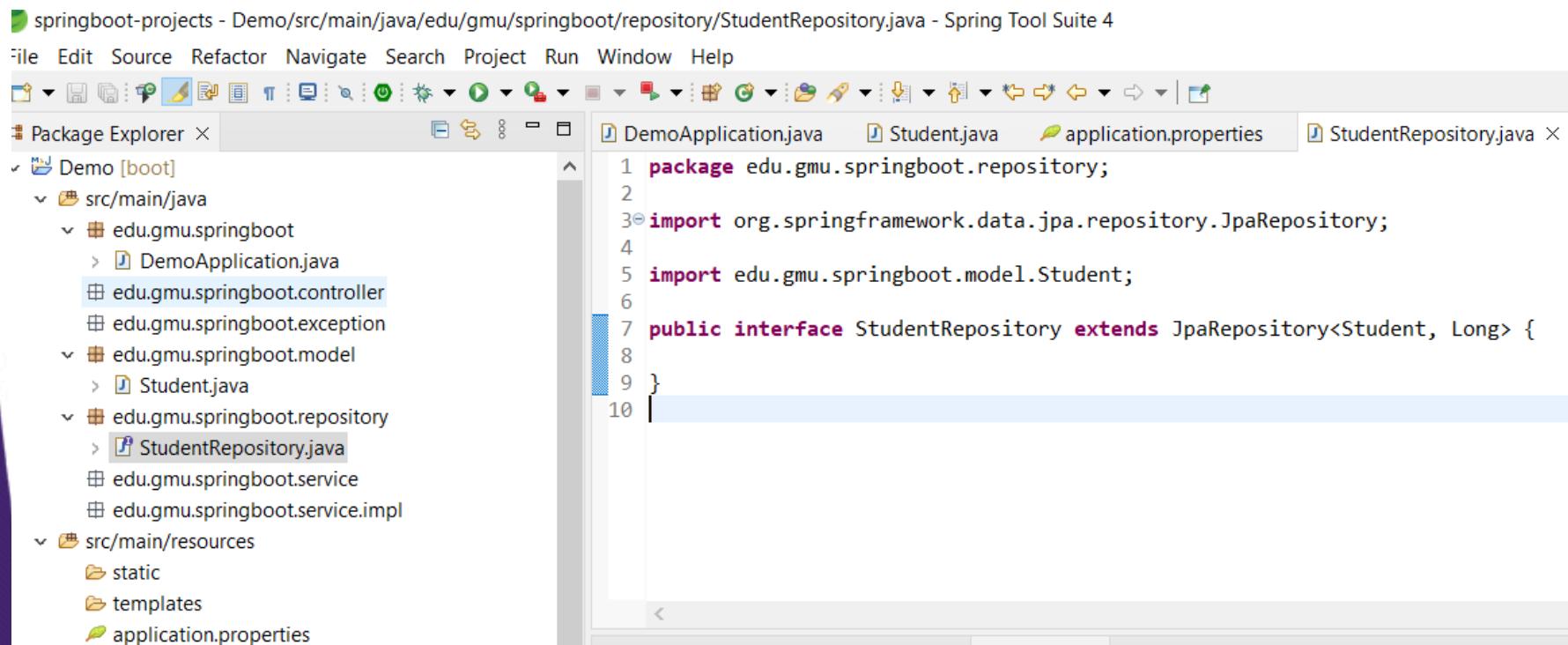
- Demo [boot]
- src/main/java
 - edu.gmu.springboot
 - DemoApplication.java
 - controller
 - exception
 - model
 - Student.java
 - repository
 - StudentRepository.java
 - service
 - service.impl
 - static
 - templates
 - application.properties

The right side shows the code editor with the file "StudentRepository.java" open. The code is as follows:

```
1 package edu.gmu.springboot.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import edu.gmu.springboot.model.Student;
6
7 public interface StudentRepository extends JpaRepository<Student, Long> {
8
9 }
10 }
```

Create Student Repository Interface

- *No need to add `@Repository` annotation to `StudentRepository` interface since Spring Data JPA takes care of this automatically.*
- *Also, no need to add `@Transaction` annotation to `StudentServiceImpl` class as Spring Data JPA internally provides transaction support to all of `StudentServiceImpl` class methods*



The screenshot shows the Spring Tool Suite 4 interface. The top bar displays the title "springboot-projects - Demo/src/main/java/edu/gmu/springboot/repository/StudentRepository.java - Spring Tool Suite 4" and a menu bar with File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. Below the menu is a toolbar with various icons. On the left is the "Package Explorer" view, which shows the project structure under "Demo [boot]". The "src/main/java" folder contains packages like "edu.gmu.springboot", "edu.gmu.springboot.controller", "edu.gmu.springboot.exception", "edu.gmu.springboot.model", "edu.gmu.springboot.repository", and "edu.gmu.springboot.service.impl". The "src/main/resources" folder contains "static", "templates", and "application.properties". On the right is the "Code Editor" view, which is currently displaying the "StudentRepository.java" file. The code is as follows:

```
1 package edu.gmu.springboot.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import edu.gmu.springboot.model.Student;
6
7 public interface StudentRepository extends JpaRepository<Student, Long> {
8
9 }
10
```

ResourceNotFoundException

Lets create a custom resource not found exception in exception package

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure under "Demo [boot]". The "src/main/java" folder contains packages: edu.gmu.springboot (with DemoApplication.java), edu.gmu.springboot.controller, edu.gmu.springboot.exception (with ResourceNotFoundException.java), edu.gmu.springboot.model (with Student.java), edu.gmu.springboot.repository (with StudentRepository.java), and edu.gmu.springboot.service (with StudentService.java and its implementation StudentServiceImpl.java). The "src/main/resources" folder contains static resources and application.properties.
- DemoApplication.java:** The main application class.
- ResourceNotFoundException.java:** The custom exception class being edited. It extends RuntimeException and includes fields for resourceName, fieldName, and fieldValue, along with their corresponding getters and setters.
- Boot Dashboard:** A central dashboard for managing projects and tasks.

```
1 package edu.gmu.springboot.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 @ResponseStatus(value = HttpStatus.NOT_FOUND)
7 public class ResourceNotFoundException extends RuntimeException{
8     /**
9      *
10     */
11    private static final long serialVersionUID = 1L;
12    private String resourceName;
13    private String fieldName;
14    private Object fieldValue;
15
16    public ResourceNotFoundException(String resourceName, String fieldName, Object fieldValue) {
17        super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));
18        this.resourceName = resourceName;
19        this.fieldName = fieldName;
20        this.fieldValue = fieldValue;
21    }
22
23    public String getResourceName() {
24        return resourceName;
25    }
26
27    public void setResourceName(String resourceName) {
28        this.resourceName = resourceName;
29    }
30
31    public String getFieldName() {
32        return fieldName;
33    }
34
35    public void setFieldName(String fieldName) {
36        this.fieldName = fieldName;
37    }
}
```

ResourceNotFoundException

Lets create a custom resource not found exception in exception package

```
1 package edu.gmu.springboot.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 @ResponseStatus(value = HttpStatus.NOT_FOUND)
7 public class ResourceNotFoundException extends RuntimeException{
8     /**
9      *
10     */
11    private static final long serialVersionUID = 1L;
12    private String resourceName;
13    private String fieldName;
14    private Object fieldValue;
15
16    public ResourceNotFoundException(String resourceName, String fieldName, Object fieldValue) {
17        super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));
18        this.resourceName = resourceName;
19        this.fieldName = fieldName;
20        this.fieldValue = fieldValue;
21    }
22
23    public String getResourceName() {
24        return resourceName;
25    }
26
27    public void setResourceName(String resourceName) {
28        this.resourceName = resourceName;
29    }
30
31    public String getFieldName() {
32        return fieldName;
33    }
34
35    public void setFieldName(String fieldName) {
36        this.fieldName = fieldName;
37    }
38
39    public Object getFieldValue() {
40        return fieldValue;
41    }
42
43    public void setFieldValue(Object fieldValue) {
44        this.fieldValue = fieldValue;
45    }
46
47
48 }
```

Service Layer Implementation

Controller layer depends on Service layer

Create `StudentService` interface inside `edu.gmu.springboot.service` package

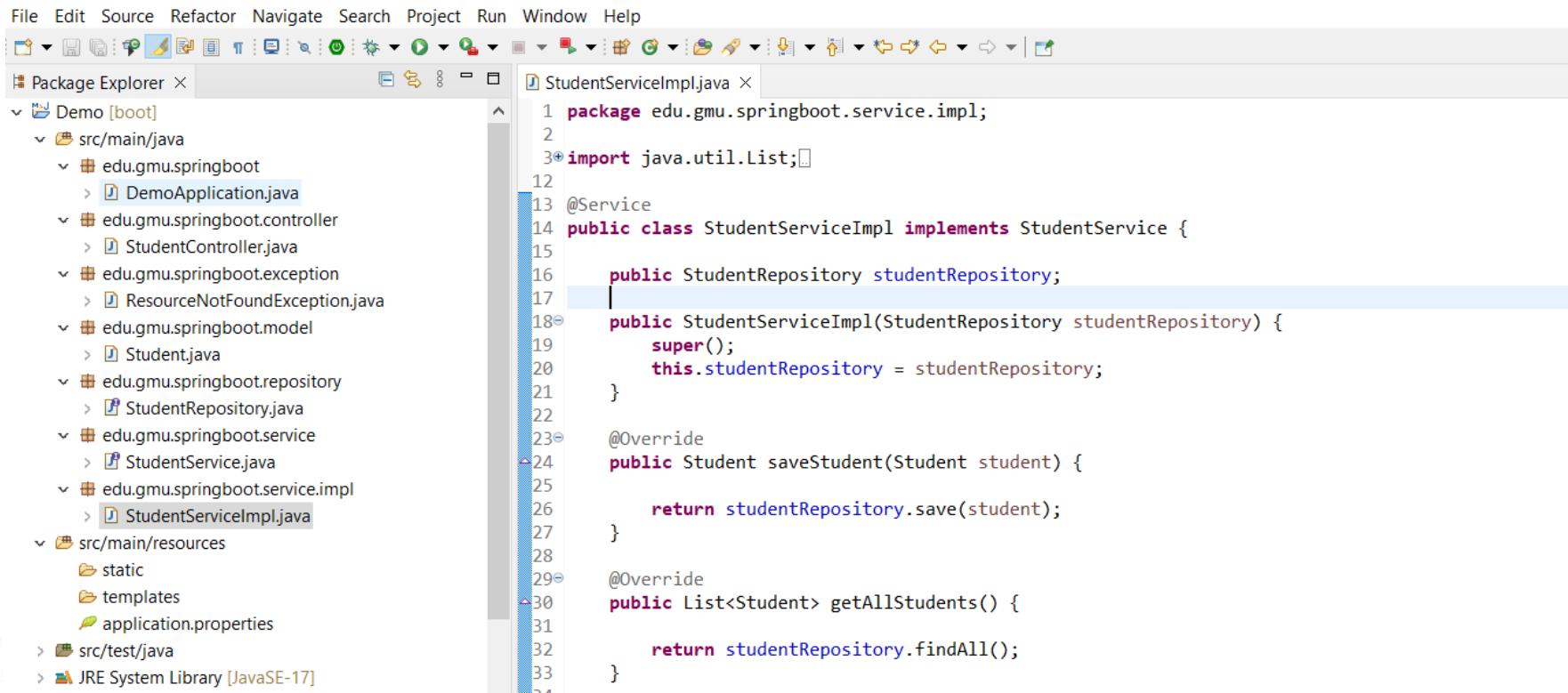
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under the 'Demo [boot]' project. It includes packages for main/java (edu.gmu.springboot, edu.gmu.springboot.controller, edu.gmu.springboot.exception, edu.gmu.springboot.model, edu.gmu.springboot.repository, edu.gmu.springboot.service), test/java, JRE System Library (JavaSE-17), Maven Dependencies, and src. The 'edu.gmu.springboot.service' package contains a file named 'StudentService.java'. On the right, the code editor window shows the contents of 'StudentService.java':

```
1 package edu.gmu.springboot.service;
2
3 import java.util.List;
4
5 import edu.gmu.springboot.model.Student;
6
7 public interface StudentService {
8     Student saveStudent(Student student);
9     List<Student> getAllStudents();
10    Student getStudentById(long id);
11    Student updateStudent(Student student, long id);
12    void deleteStudent(long id);
13 }
14
15 }
```

Service Layer Implementation

Controller layer depends on Service layer

Create `StudentServiceImpl` class that implements `StudentService`



The screenshot shows the Eclipse IDE interface. The left side features the 'Package Explorer' view, which displays the project structure of 'Demo [boot]'. The 'src/main/java' package contains several sub-packages and files, including 'edu.gmu.springboot' (with 'DemoApplication.java'), 'controller' (with 'StudentController.java'), 'exception' (with 'ResourceNotFoundException.java'), 'model' (with 'Student.java'), 'repository' (with 'StudentRepository.java'), 'service' (with 'StudentService.java'), and 'service.impl' (with 'StudentServiceImpl.java'). The right side shows the code editor for 'StudentServiceImpl.java'. The code implements the 'StudentService' interface, utilizing 'StudentRepository' to perform operations on 'Student' objects.

```
1 package edu.gmu.springboot.service.impl;
2
3 import java.util.List;
4
5 @Service
6 public class StudentServiceImpl implements StudentService {
7
8     public StudentRepository studentRepository;
9
10    public StudentServiceImpl(StudentRepository studentRepository) {
11        super();
12        this.studentRepository = studentRepository;
13    }
14
15    @Override
16    public Student saveStudent(Student student) {
17
18        return studentRepository.save(student);
19    }
20
21    @Override
22    public List<Student> getAllStudents() {
23
24        return studentRepository.findAll();
25    }
26
27
28
29
30
31
32
33 }
```

Service Layer Implementation

Controller layer depends on Service layer

Create StudentServiceImpl interface that implements StudentService

```
StudentServiceImpl.java ×
34
35@override
36public Student getStudentById(long id) {
37    //Please note that repository's findById() method returns a Optional object (of java.util)
38    //which is a generic, hence we need to pass the type Student to the returned Optional object
39    Optional<Student> student = studentRepository.findById(id);
40
41    //Check if the returned Optional object contains the student, if so, return the student object
42    if (student.isPresent()){
43        return student.get(); // here get() method returns the content of the Optional object.
44    }else {
45        //this internally creates a message. Check the ResourceNotFoundException for details.
46        throw new ResourceNotFoundException("Student", "Id", id);
47    }
48
49    //Above code can be replaced by one line code with lambda expression
50    // return studentRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Employee", "Id", id));
51}
52
53@Override
54public Student updateStudent(Student student, long id) {
55    //check whether student with given id exists in database
56    Student existingStudent = studentRepository.findById(id).orElseThrow(
57        () -> new ResourceNotFoundException("Student", "Id", id));
58
59    existingStudent.setFirstName(student.getFirstName());
60    existingStudent.setLastName(student.getLastName());
61    existingStudent.setEmail(student.getEmail());
62
63    //save existing student to database
64    studentRepository.save(existingStudent);
65
66    return existingStudent; //return the existing student to the controller layer
67}
68
69@Override
70public void deleteStudent(long id) {
71    //check whether the student exists in the database
72    studentRepository.findById(id).orElseThrow( () -> new ResourceNotFoundException("Student", "Id", id));
73
74    studentRepository.deleteById(id);|
75}
76}
```

Controller Layer Implementation

- Let's **Create StudentController** under `edu.gmu.springboot.controller` and **annotate it with `@RestController`,**
 - This is a convenient annotation that combines `@Controller` and `@ResponseBody`
 - This eliminates the need to annotate every request handling method of the controller class with the `@ResponseBody`
- **Also, inject StudentService dependency**
 - Need to inject `StudentService` using constructor based dependency injection
- **Use `@RequestMapping` annotation to provide base url for the controller (StudentController)**

StudentController

```
1 package edu.gmu.springboot.controller;
2
3 import java.util.List;
4
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.DeleteMapping;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.PutMapping;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RestController;
15
16 import edu.gmu.springboot.model.Student;
17 import edu.gmu.springboot.service.StudentService;
18
19 @RestController
20 @RequestMapping("/api/students") //defines the base url for all apis
21 public class StudentController {
22     private StudentService studentService;
23
24     //inject studentService object using constructor based dependency injection
25     public StudentController(StudentService studentService) {
26         super();
27         this.studentService = studentService;
28     }
29 }
```

StudentController: saveStudent()

- Now let's build create student REST API, saveStudent() to save Student data to the database. **Key Annotations:**
- **@PostMapping**
 - Annotating saveStudent() with @PostMapping allows this REST API (method) to handle http POST requests.
- **@RequestBody**
 - @RequestBody annotation allows to retrieve the request's body and automatically convert it to Java object
- **ResponseType**
 - Return type ResponseEntity is used to return response for the rest api
 - ResponseEntity allows to provide complete response details in this class, including status, header.
 - This api created the resource, hence we pass HttpStatus.CREATED
- **The url to access above method will be
<http://localhost:8080/api/students>**
 - when invoked with http POST method and also send data for request body, such as json object with key value pairs.
 - You can use Postman Rest client to test the RESTful APIs

StudentController: saveStudent()

```
19 @RestController
20 @RequestMapping("/api/students") //defines the base url for all apis
21 public class StudentController {
22     private StudentService studentService;
23
24     //inject studentService object using constructor based dependency injection
25     public StudentController(StudentService studentService) {
26         super();
27         this.studentService = studentService;
28     }
29
30     //Build create student api that handles post requests
31
32     @PostMapping
33     public ResponseEntity<Student> saveStudent(@RequestBody Student student) {
34         return new ResponseEntity<Student>(studentService.saveStudent(student), HttpStatus.CREATED);
35     }
36
```

StudentController: getAllStudents()

- Now let's create getAll Students() REST API to get all students from the database table,
 - getAllStudents() REST API within the StudentController.
 - Since StudentController depends on StudentService, so add getAllStudents() method in StudentService interface, and implement the getAllStudents() method in StudentServiceImpl class.
- **public List<Student> getAllStudents() rest api can be accessed using the url**
 - <http://localhost:8080/api/students>, which by default uses http GET method.
 - This may return array of JSON objects.

StudentController: getAllStudents()

```
19 @RestController
20 @RequestMapping("/api/students") //defines the base url for all apis
21 public class StudentController {
22     private StudentService studentService;
23
24     //inject studentService object using constructor based dependency injection
25     public StudentController(StudentService studentService) {
26         super();
27         this.studentService = studentService;
28     }
29
30     //Build create student api that handles post requests
31
32     @PostMapping
33     public ResponseEntity<Student> saveStudent(@RequestBody Student student) {
34         return new ResponseEntity<Student>(studentService.saveStudent(student), HttpStatus.CREATED);
35     }
36
37     //Create get all students REST API
38     @GetMapping
39     public List<Student> getAllStudents(){
40         return studentService.getAllStudents();
41     }
42
```

StudentController: getStuedentById()

- Now, let's build a REST API that returns a specific Student for a given ID
 - Since the controller layer depends on Service layer, so need to add the corresponding method in StudentService
- In the next example {id} is url template variable
 - a syntax to get path variable which is dynamic and appended to the end of URI, e.g., 1 at the end of the following URI; allows url to be <http://localhost:8080/api/employees/1>

StudentController: getStuedentById()

```
StudentController.java ×
19 @RestController
20 @RequestMapping("/api/students") //defines the base url for all apis
21 public class StudentController {
22     private StudentService studentService;
23
24     //inject studentService object using constructor based dependency injection
25     public StudentController(StudentService studentService) {
26         super();
27         this.studentService = studentService;
28     }
29
30     //Build create student api that handles post requests
31     [
32     @PostMapping
33     public ResponseEntity<Student> saveStudent(@RequestBody Student student) {
34         return new ResponseEntity<Student>(studentService.saveStudent(student), HttpStatus.CREATED);
35     }
36
37     //Create get all students REST API
38     @GetMapping
39     public List<Student> getAllStudents(){
40         return studentService.getAllStudents();
41     }
42
43     //create get student by id REST API.
44
45     @GetMapping("{id}")
46     public ResponseEntity<Student> getStudentById(@PathVariable("id") long studentId){
47         return new ResponseEntity<Student>(studentService.getStudentById(studentId), HttpStatus.OK);
48     }
49 }
```

StudentController: updateStuedent()

- **Now to implement update Student REST API**
 - Since the Controller layer depends on Service Layer, need to update service layer first
 - So, add new method updateStudent() in StudentService interface and implment it into StudentServiceImpl
- **@PutMapping annotation makes this method as a RESTful API that can handle http PUT requests.**
 - "{id}" used with PutMapping is a syntax to bind with path variable value
 - client can use URL of sort
`http://localhost:8080/api/employees/1`
- **Return an instance of ResponseEntity class with two parameters- body and status**

StudentController: updateStuedent()

```
50
51     //Now to implement update Student REST API
52
53     @PutMapping("{id}")
54     public ResponseEntity<Student> updateStudent(@PathVariable("id") long id, @RequestBody Student student){
55         return new ResponseEntity<Student>(studentService.updateStudent(student, id), HttpStatus.OK);
56     }
57
58     // Create delete student REST API.
59
60     @DeleteMapping("{id}")
61     public ResponseEntity<String> deleteStudent(@PathVariable("id") long id){
62         //delete student from database
63         studentService.deleteStudent(id);
64
65         return new ResponseEntity<String>("Employee deleted successfully.", HttpStatus.OK);
66     }
67
68 }
69 }
```

StudentController: deleteStuedent()

- **Finally let's create delete student REST API.**
 - Need to make change in the service layer first since the controller layer depends on Service layer
- **This method becomes REST api by using the annotation @DeleteMapping**
- **API can be accessed using the following URL using http method delete**
 - `http://localhost:8080/api/students/1`

StudentController: deleteStuedent()

```
50
51     //Now to implement update Student REST API
52
53     @PutMapping("{id}")
54     public ResponseEntity<Student> updateStudent(@PathVariable("id") long id, @RequestBody Student student){
55         return new ResponseEntity<Student>(studentService.updateStudent(student, id), HttpStatus.OK);
56     }
57
58     // Create delete student REST API.
59
60     @DeleteMapping("{id}")
61     public ResponseEntity<String> deleteStudent(@PathVariable("id") long id){
62         //delete student from database
63         studentService.deleteStudent(id);
64
65         return new ResponseEntity<String>("Employee deleted successfully.", HttpStatus.OK);
66     }
67
68 }
69 }
```

Installing MySQL Server and Workbench on Windows 10

Installing MySQL Server and Workbench on Windows 10

Google search
mysql and
select the
first link.

A screenshot of a Google search results page. The search bar at the top contains the query "mysql". Below the search bar, there are tabs for "All", "Books", "Videos", "Images", "News", and "More", with "All" being the selected tab. To the right of these tabs is a "Tools" link. The search results indicate "About 1,730,000,000 results (0.64 seconds)". The top result is a link to the MySQL website: <https://www.mysql.com>. The page title is "MySQL" followed by a green checkmark icon. The snippet below the title reads: "MySQL Database Service is a fully managed database service to deploy cloud-native applications. HeatWave, an integrated, high-performance query accelerator ...". Below the main search results, there is a section titled "Results from mysql.com" containing links to "Downloads" (with a green checkmark), "Documentation" (with a green checkmark), "MySQL Community Downloads" (with a green checkmark), and "Developer Zone" (with a green checkmark). Each of these sections has a brief description and a link to the corresponding page on the MySQL website.

mysql

All Books Videos Images News More Tools

About 1,730,000,000 results (0.64 seconds)

<https://www.mysql.com> :: MySQL ✓

MySQL Database Service is a fully managed database service to deploy cloud-native applications. HeatWave, an integrated, high-performance query accelerator ...

Results from mysql.com

Downloads ✓

MySQL Community - MySQL
Installer 8.0.27 - Downloads - ...

Documentation ✓

MySQL Workbench - MySQL
8.0.27 - MySQL 8.0 Release Notes

MySQL Community Downloads ✓

MySQL Installer 8.0.27 - Download
MySQL - Connector/J 8.0.27 - ...

Developer Zone ✓

MySQL Query Optimization is
usually simple engineering. But ...

*Google search mysql
and select the first link.*

The screenshot shows the official MySQL website. At the top, there's a navigation bar with links for MySQL.COM (which is highlighted with an orange underline), DOWNLOADS, DOCUMENTATION, and DEVELOPER ZONE. Below this is a secondary navigation bar with links for Products, Cloud, Services, Partners, Customers, Why MySQL?, News & Events, and How to Buy. A search bar is located at the top right, along with 'Contact MySQL' and 'Login' buttons. On the left, the MySQL logo is displayed with its tagline 'The world's most popular open source database'. In the center, there's a banner for 'ORACLE LIVE' featuring a video thumbnail about 'MySQL Database Service—New HeatWave Innovations' by Edward Sreven, Chief Corporate Architect, Oracle. A 'WATCH NOW' button is visible below the thumbnail. The main content area contains four product cards: 'MySQL Database Service with HeatWave' (cloud icon), 'MySQL Enterprise Edition' (database icon), 'MySQL for OEM/ISV' (server icon), and 'MySQL Cluster CGE' (globe icon). Each card includes a brief description and a 'Learn More' link.

The world's most popular open source database

MySQL™

MySQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE

Contact MySQL | Login

f t

Products Cloud Services Partners Customers Why MySQL? News & Events How to Buy

ORACLE LIVE

MySQL Database Service—New HeatWave Innovations

with Edward Sreven, Chief Corporate Architect, Oracle

WATCH NOW

MySQL Database Service with HeatWave

MySQL Database Service is a fully managed database service to deploy cloud-native applications. HeatWave, an integrated, high-performance query accelerator boosts MySQL performance by 5400x.

[Learn More »](#)

MySQL Enterprise Edition

The most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.

[Learn More »](#)

MySQL for OEM/ISV

Over 2000 ISVs, OEMs, and VARs rely on MySQL as their products' embedded database to make their applications, hardware and appliances more competitive, bring them to market faster, and lower their cost of goods sold.

[Learn More »](#)

MySQL Cluster CGE

MySQL Cluster enables users to meet the database challenges of next generation web, cloud, and communications services with uncompromising scalability, uptime and agility.

[Learn More »](#)

Click on DOWNLOADS and then click on “MySQL Community



MySQL Database Service with HeatWave

Faster Performance

- 5400x Faster than Amazon RDS
- 1400x Faster than Amazon Aurora
- 6.5x Faster than Amazon Redshift AQUA
- 7x Faster than Snowflake

[Try Now](#)

Free Webinars

Getting Started with MySQL in Israel's New Data Center (in Hebrew)

Tuesday, January 25, 2022

Maximizing Query Performance for MySQL On-Premises and in the Cloud

Wednesday, January 26, 2022

Getting Started with MySQL in Abu Dhabi's New Data Center

Tuesday, February 08, 2022

[More »](#)

Contact Sales

USA: +1-866-221-0634

Canada: +1-866-221-0634

Germany: +49 89 143 01280

France: +33 1 57 60 83 57

Italy: +39 02 249 59 120

UK: +44 207 553 8447

MySQL Enterprise Edition

MySQL Enterprise Edition includes the most comprehensive set of advanced MySQL.

[Learn More »](#)

[Customer Download »](#)

[Trial Download »](#)

MySQL Cluster CGE

MySQL Cluster is a real-time open source transactional database designed for conditions.

- MySQL Cluster
- MySQL Cluster Manager
- Plus, everything in MySQL Enterprise Edition

[Learn More »](#)

[Customer Download »](#) (Select Patches & Updates Tab, Product Search)

[Trial Download »](#)

[MySQL Community \(GPL\) Downloads »](#)

On “MySQL Community Downloads” page, click on “MySQL Installer for Windows”

④ MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench

- MySQL Installer for Windows
- MySQL for Visual Studio

- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/.NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

On “MySQL Installer for Windows”, download ‘Windows (x86, 32-bit, MSI Installer (the 2nd link below) – this works for 62-bit O/S as well.

⬇ MySQL Community Downloads

◀ MySQL Installer

The screenshot shows the MySQL Community Downloads page for MySQL Installer 8.0.27. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, the title "MySQL Installer 8.0.27" is displayed. A dropdown menu "Select Operating System:" is set to "Microsoft Windows". To the right, a link says "Looking for previous GA versions?". The main content area lists two download options:

Version	File Size	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.27.1.msi)	8.0.27 2.3M	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.27.1.msi)	8.0.27 470.2M	Download

A large blue oval highlights the second download link for "Windows (x86, 32-bit), MSI Installer".

You do not need to Login or Sign up. Just Press “No thanks, just start my download.”

⬇ MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »

using my Oracle Web account

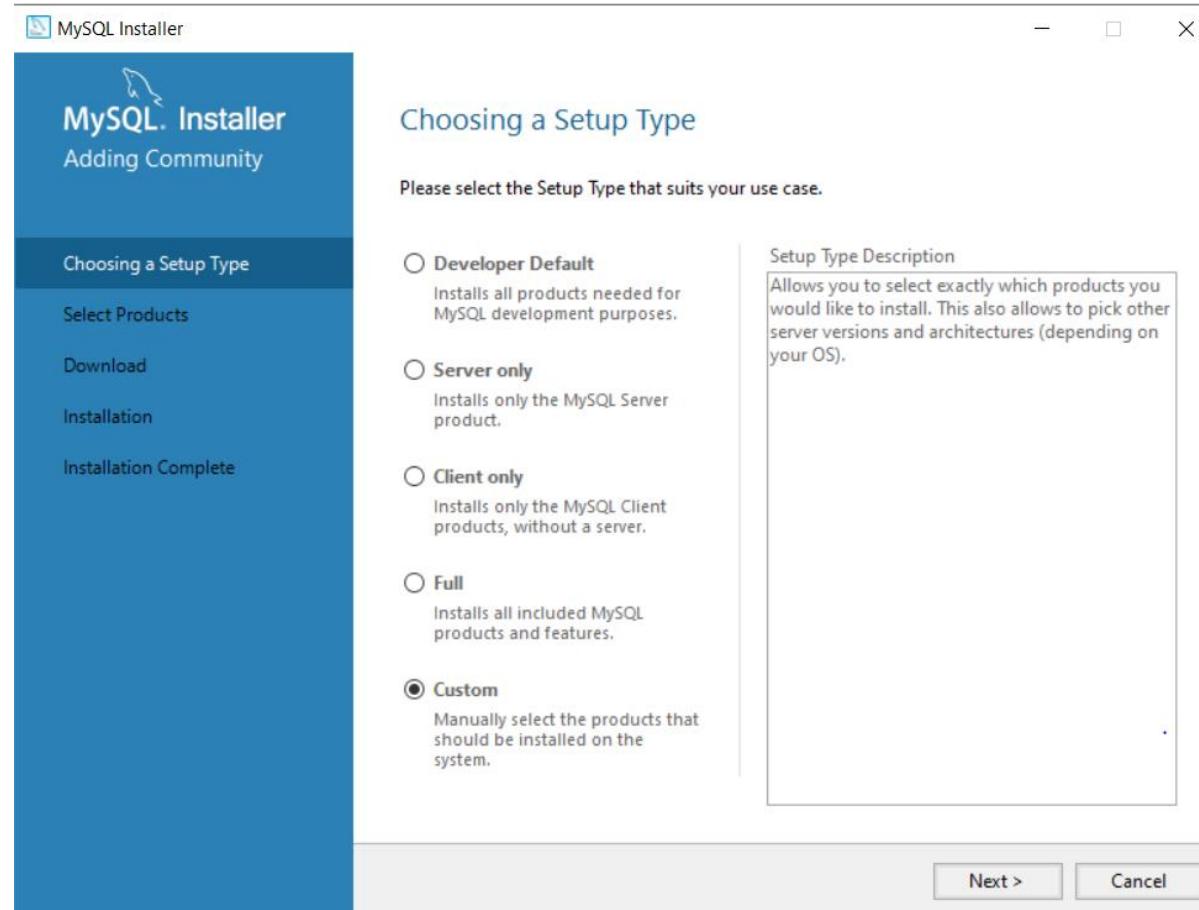
Sign Up »

for an Oracle Web account

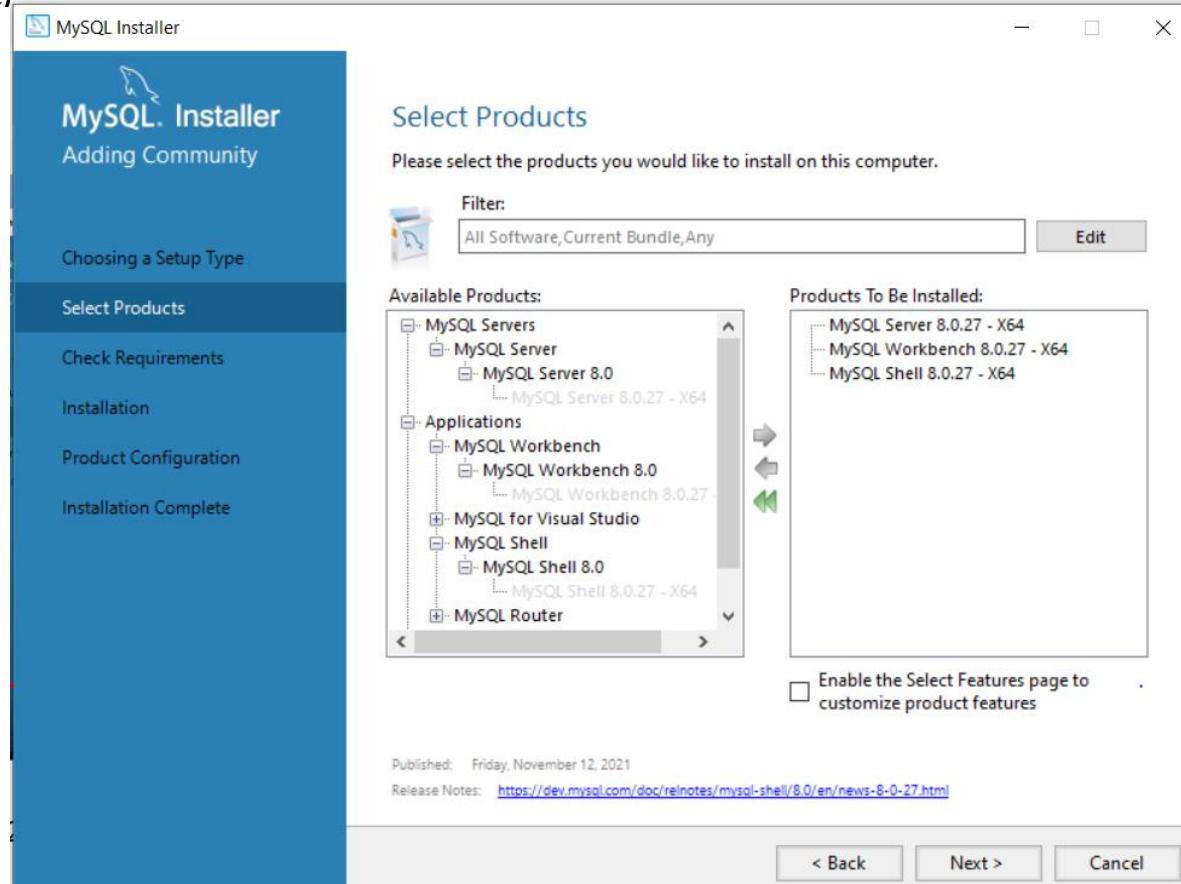
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

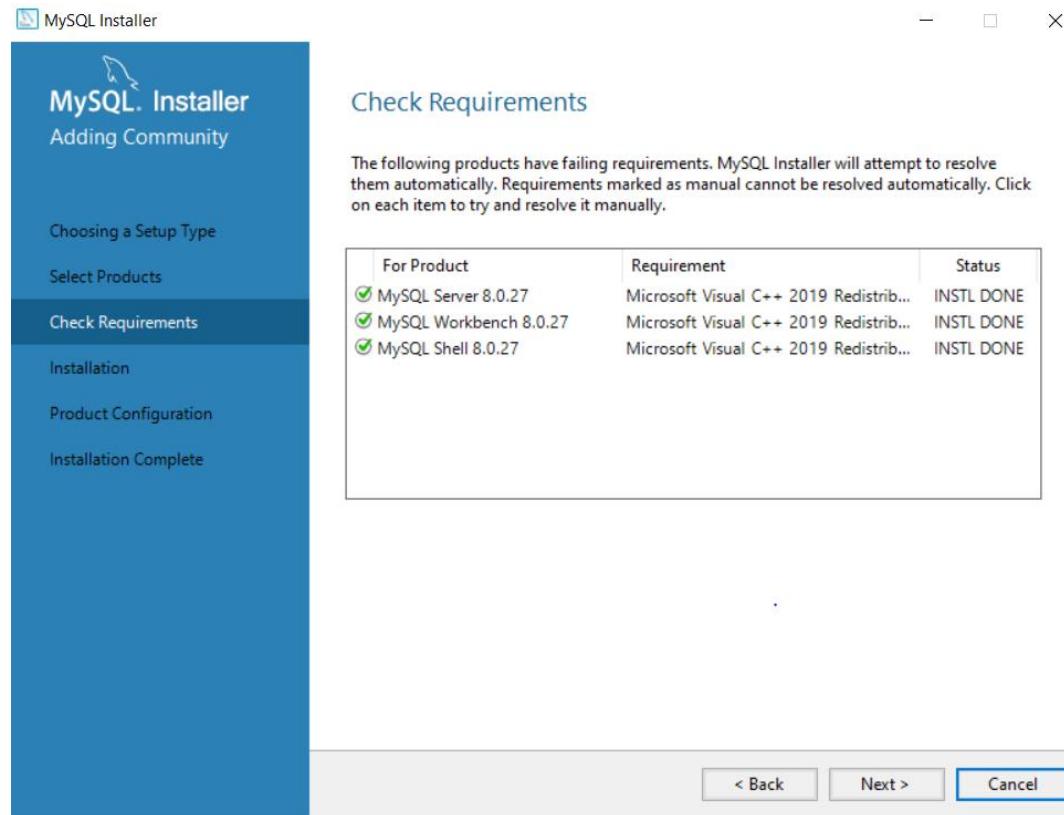
Double click on the downloaded msi file and Select Custom, press Next



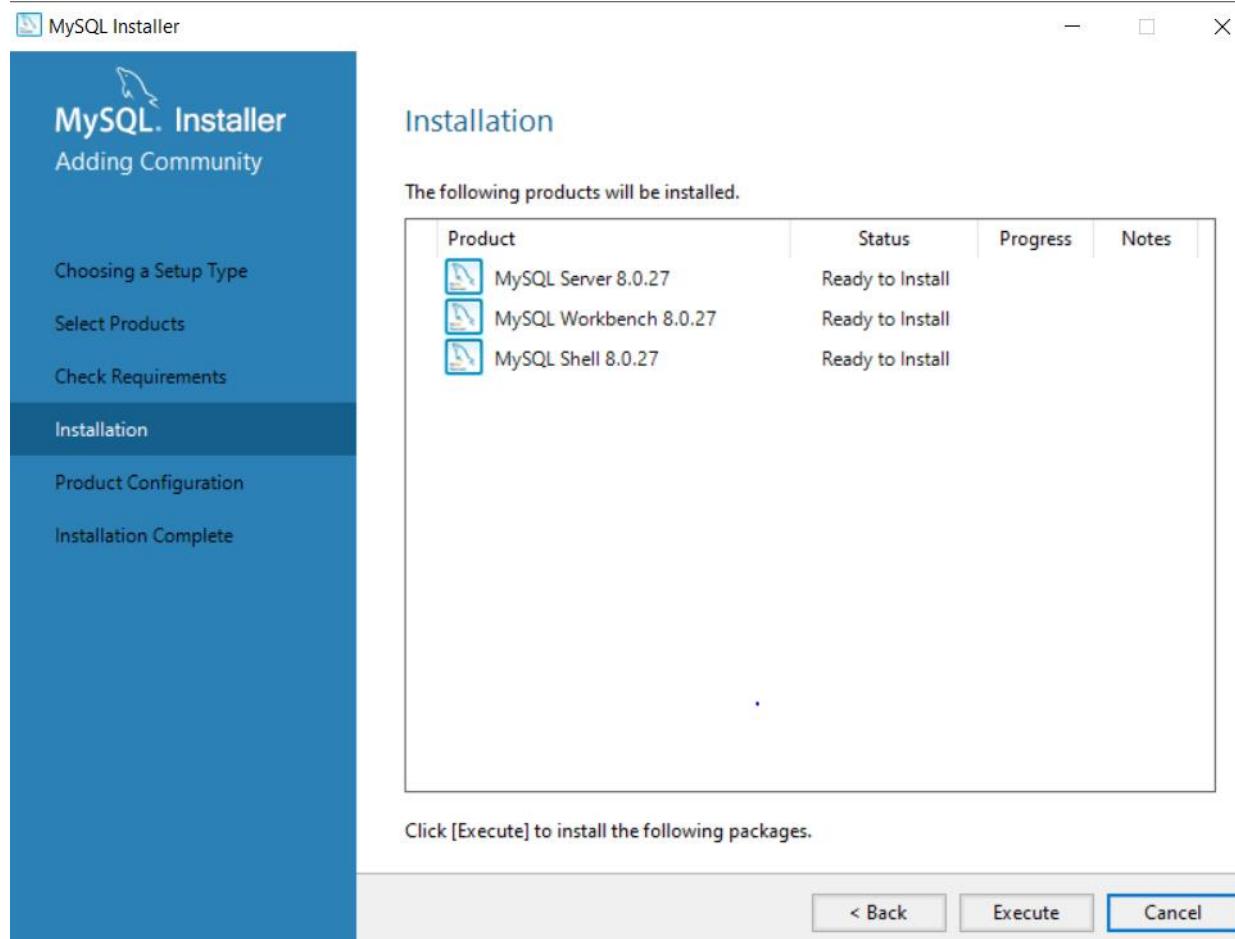
Select MySQL Server, MySQL Workbench, and MySQL Shell so that they appear in the right hand box. Press Next



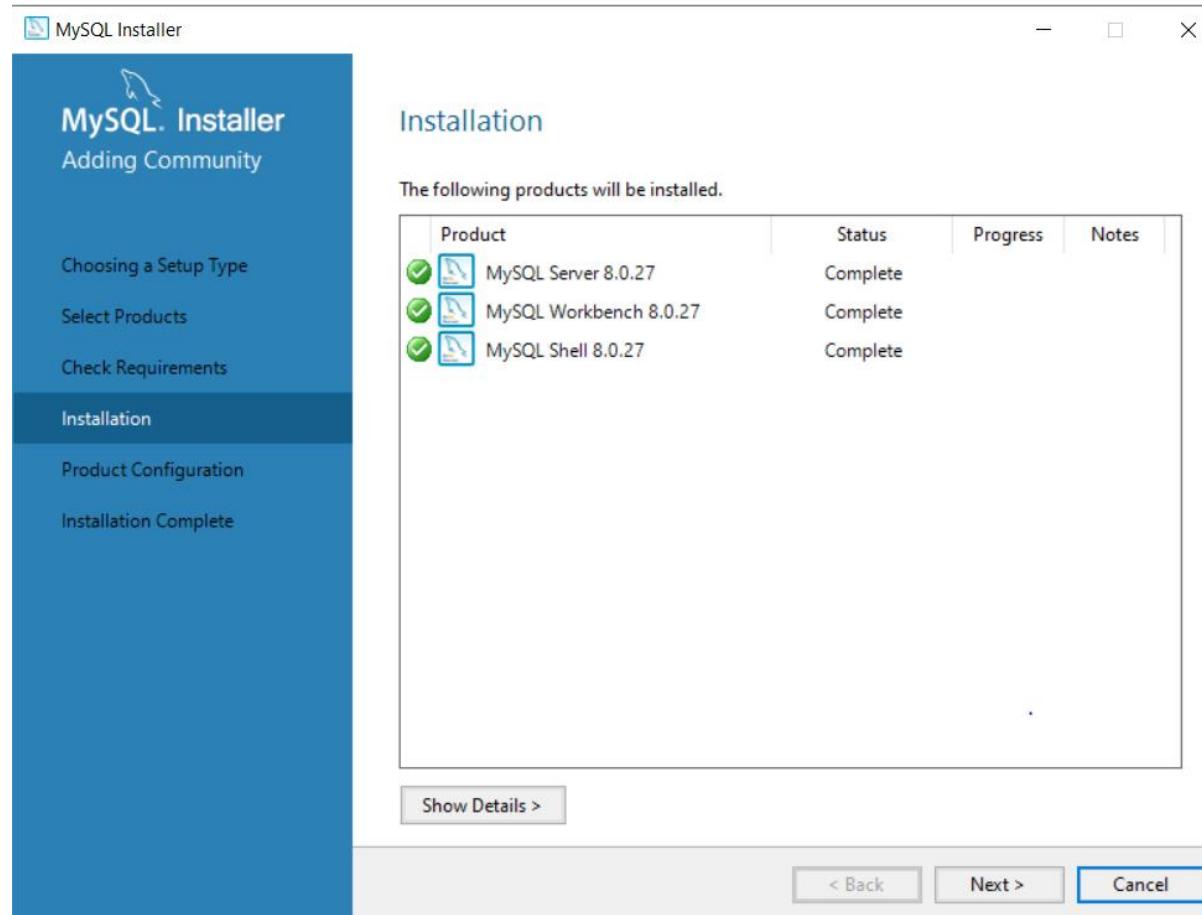
Press Next



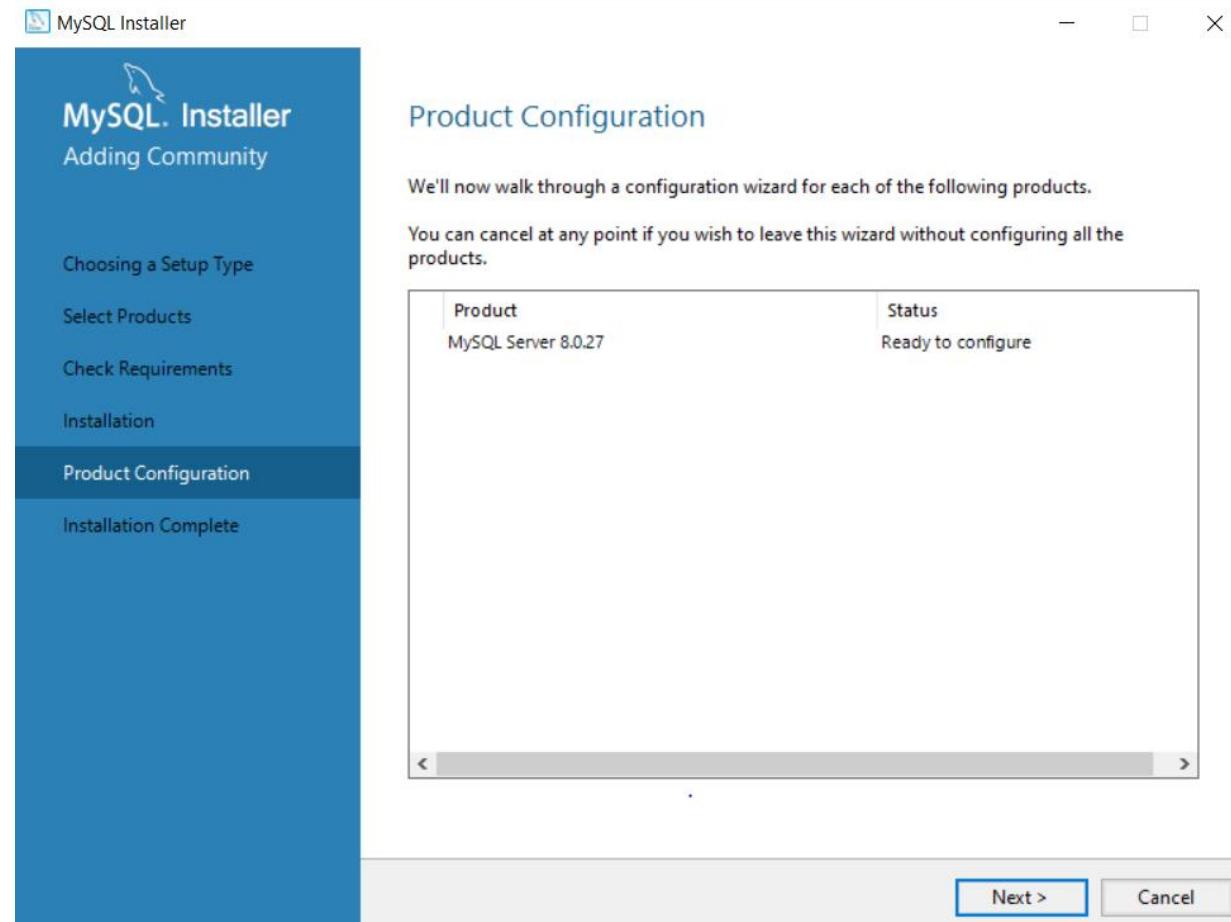
Press Execute



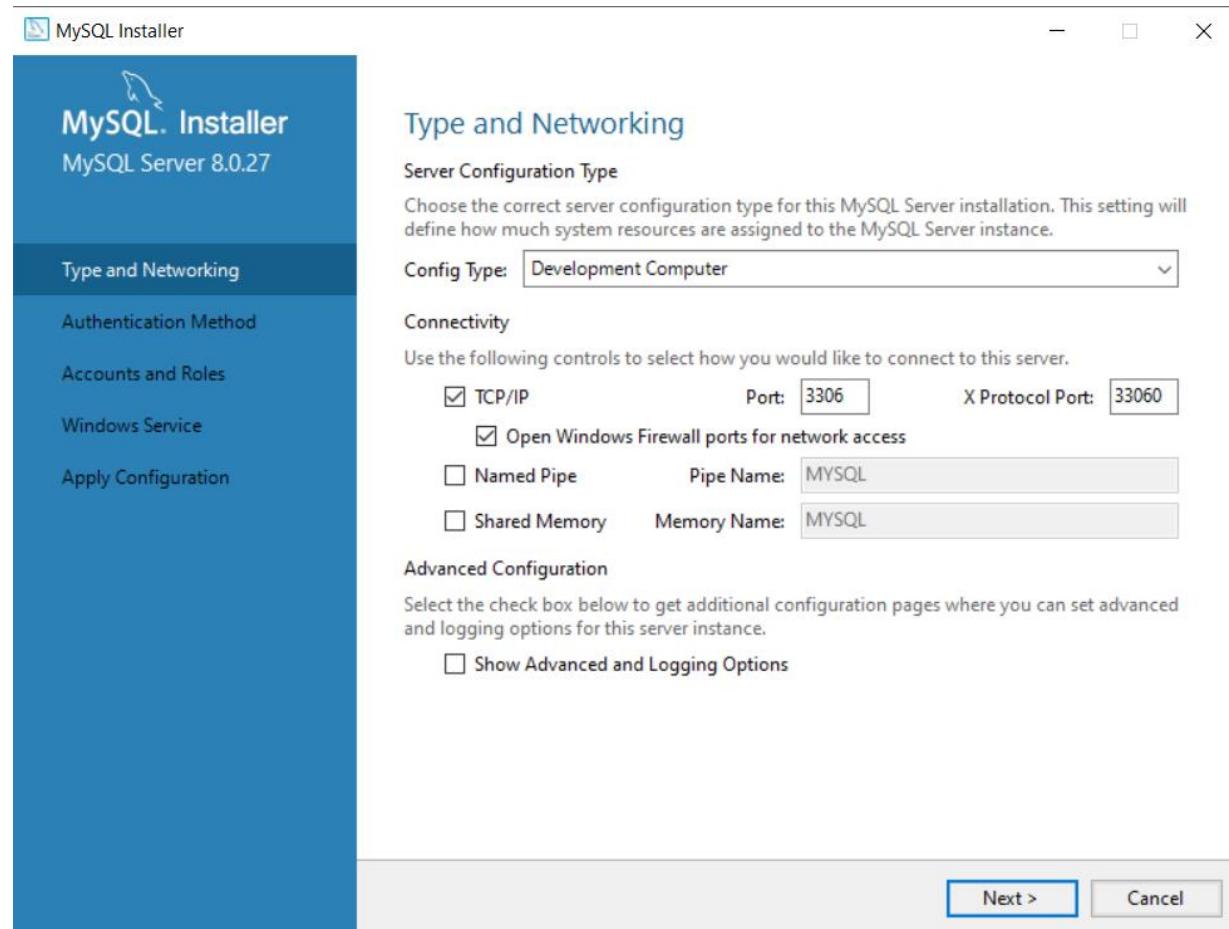
Here is how it looks when installed. Press Next



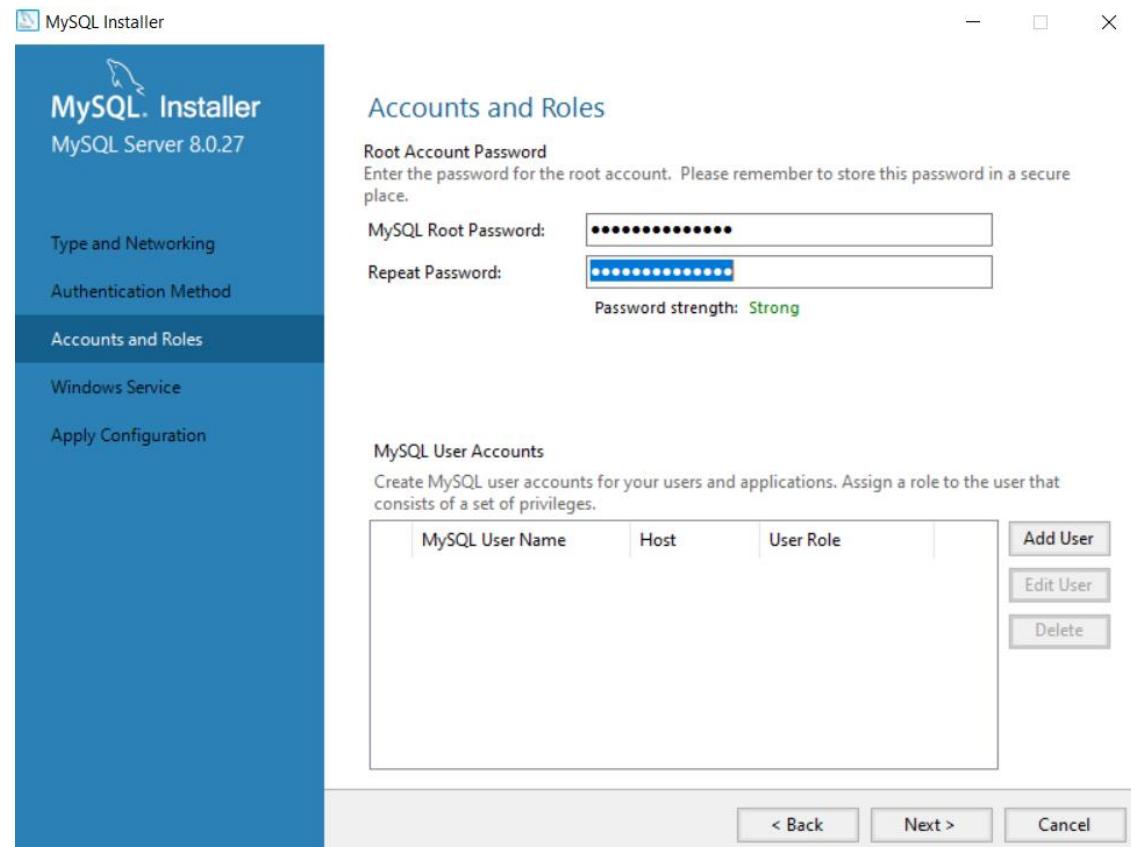
On the Product Configuration page, Press Next



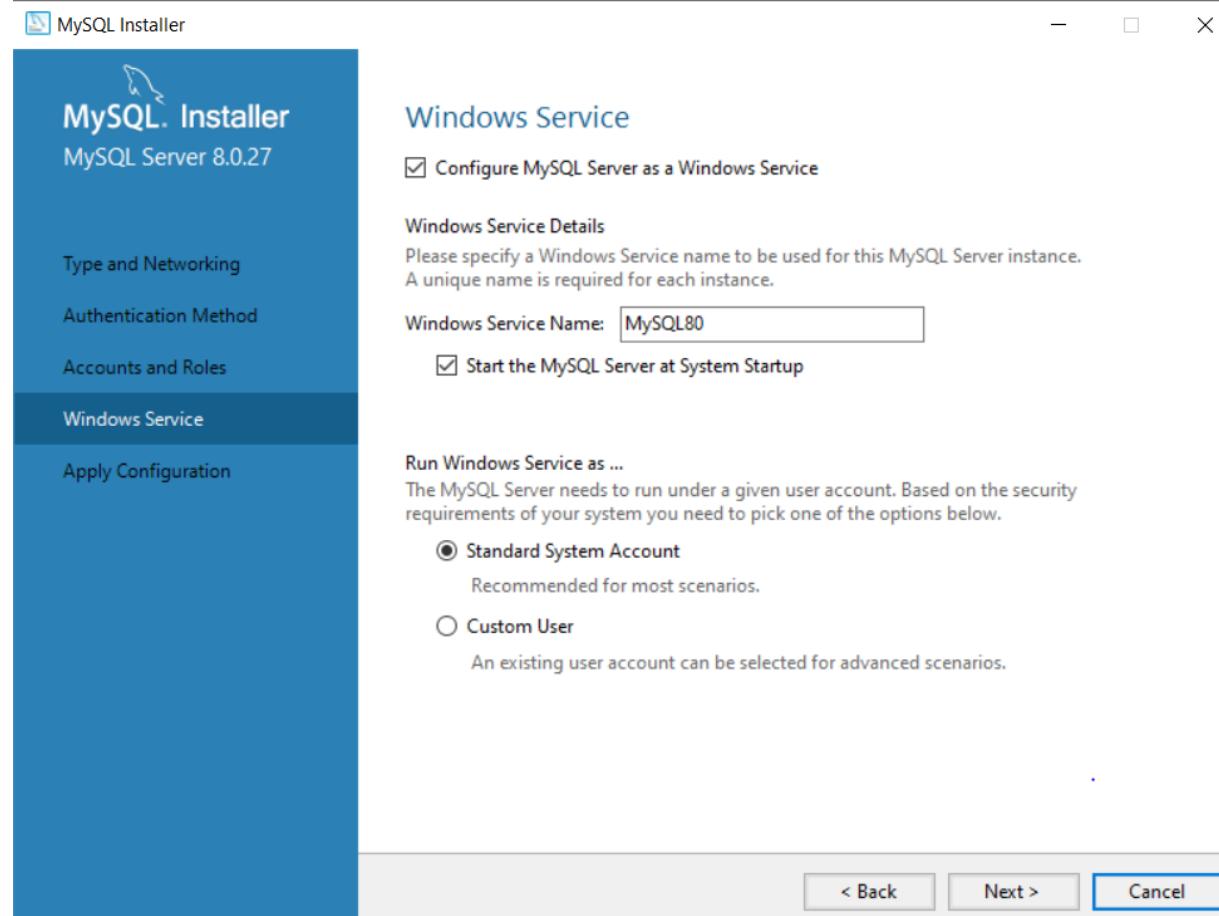
On the Type and Networking page, keep default, Press Next



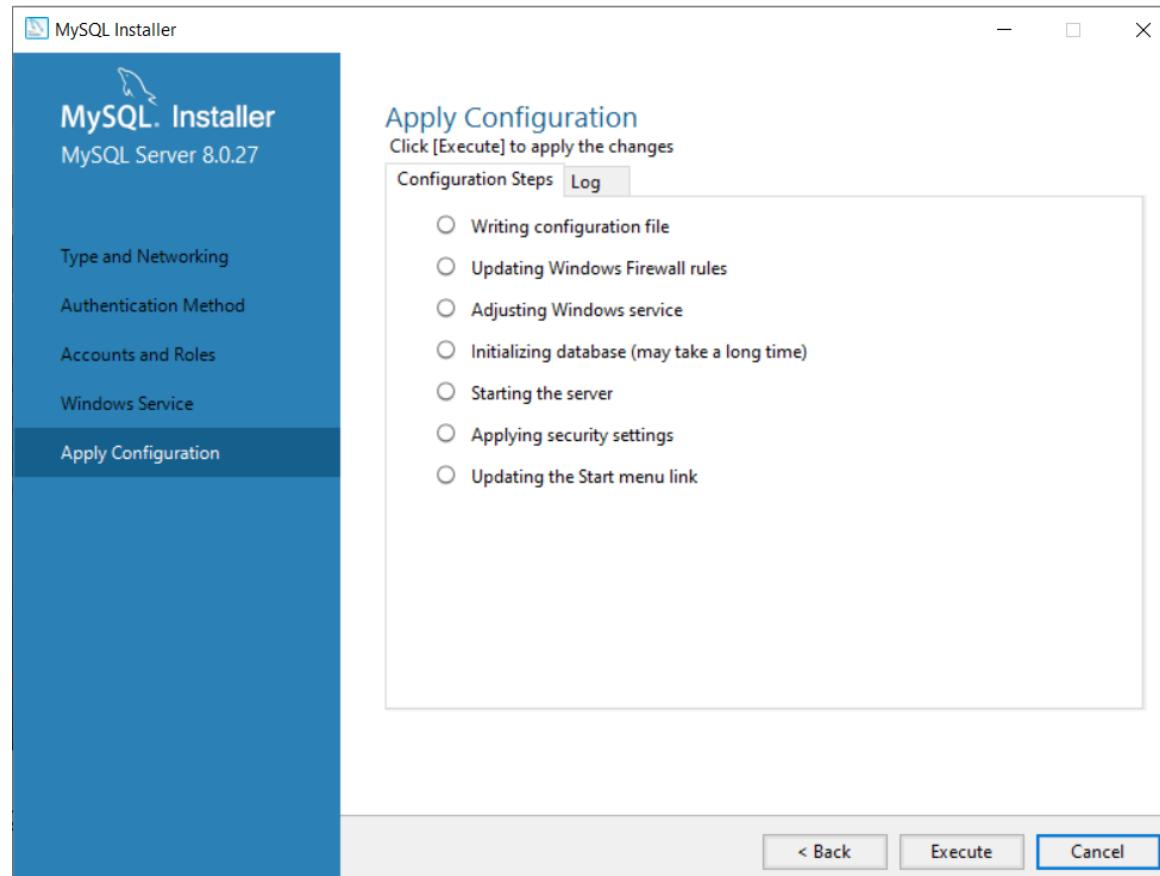
Select Strong Password for root user, Press Next



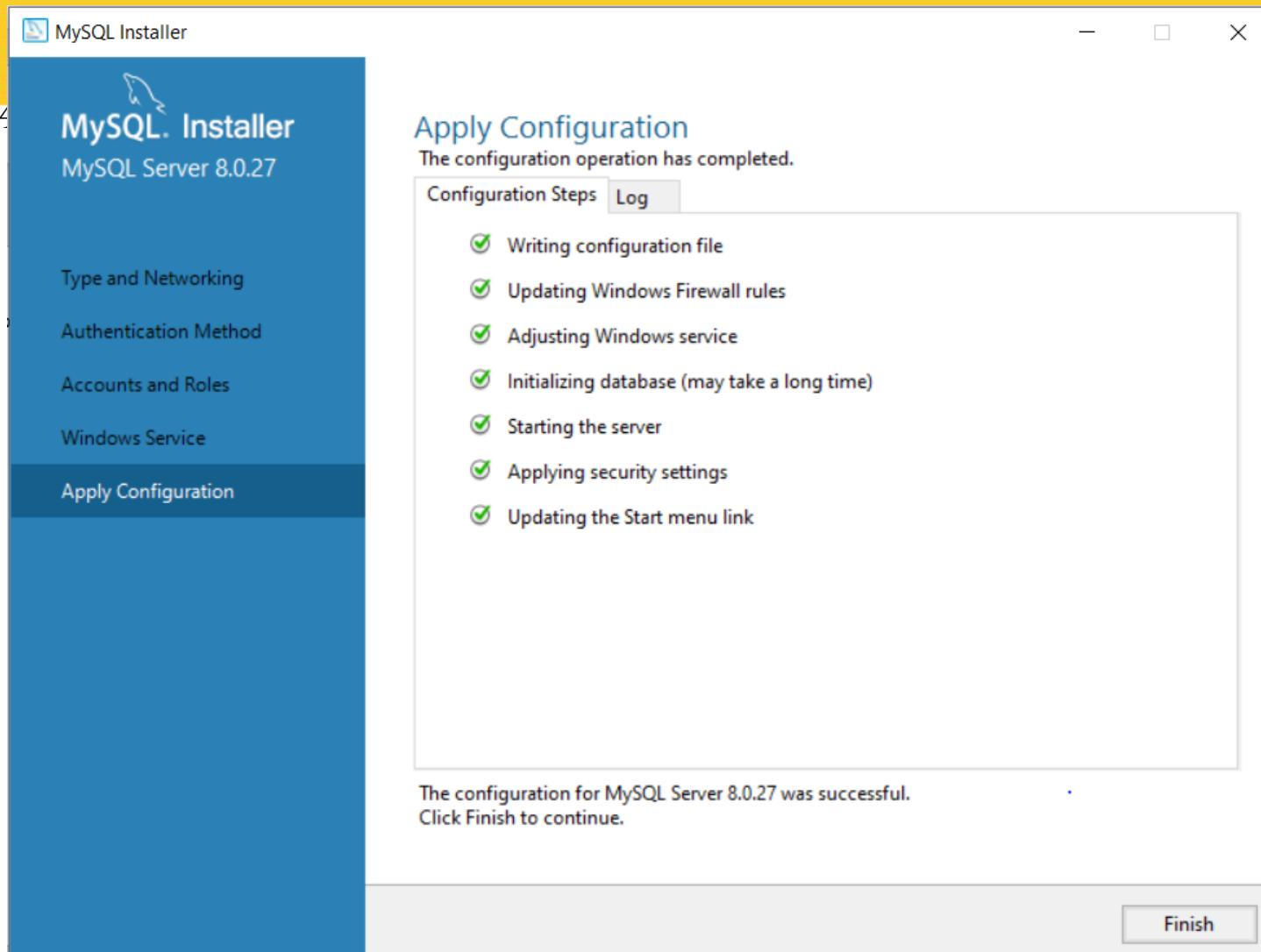
On Windows Service page, keep default, Press Next



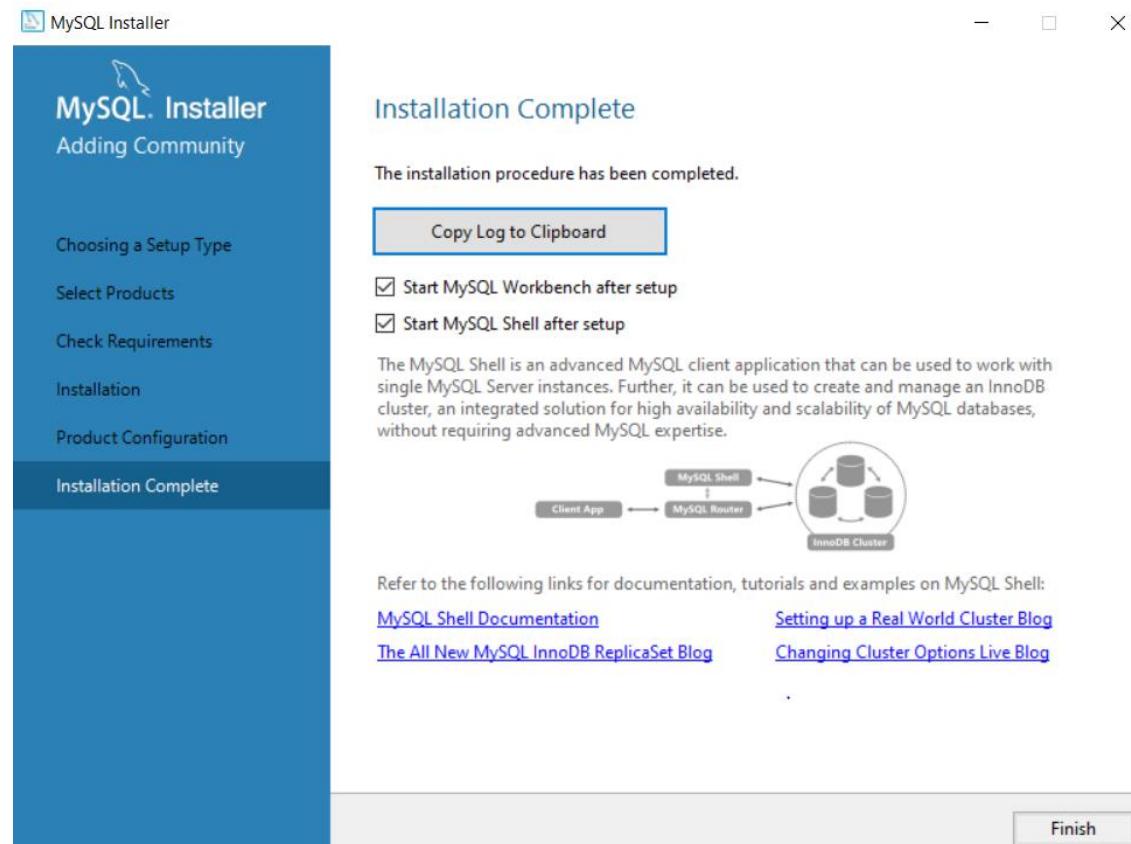
On Apply Configuration page, keep default, Press Execute



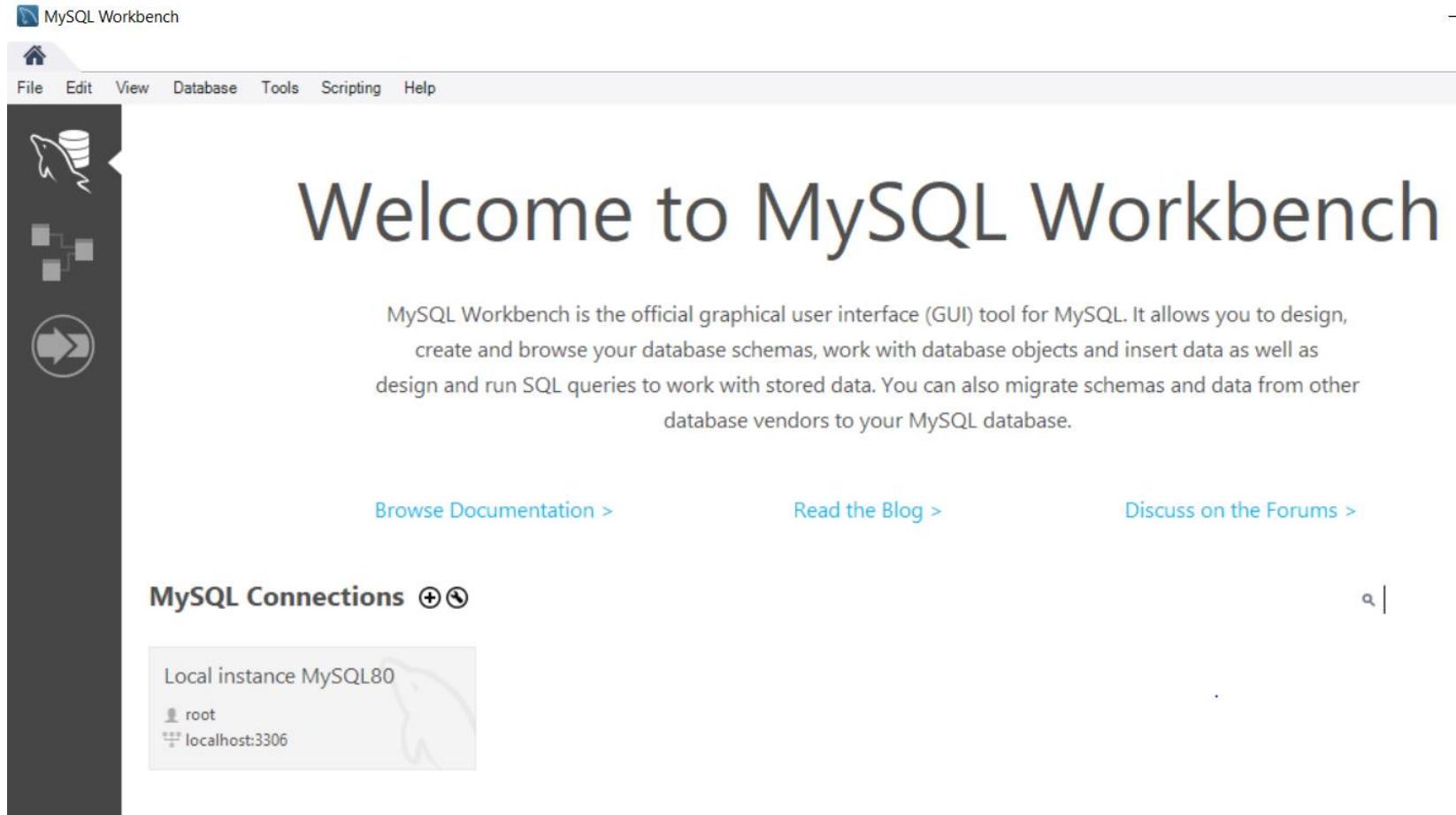
On A



Press Next and then press Finish



You will see console for MySQL Workbench opens



The screenshot shows the MySQL Workbench application window. At the top, there's a menu bar with File, Edit, View, Database, Tools, Scripting, and Help. Below the menu is a toolbar with icons for Home, New Connection, Open Connection, and Help. The main area features a large "Welcome to MySQL Workbench" title. To the left of the title is a sidebar with three icons: a MySQL logo, a schema diagram, and a circular arrow. Below the sidebar, the text describes MySQL Workbench as a graphical user interface for MySQL, mentioning its capabilities like schema design, object management, data insertion, SQL query execution, and schema migration. At the bottom of the main area are three links: "Browse Documentation >", "Read the Blog >", and "Discuss on the Forums >". The bottom right corner has a search bar with a magnifying glass icon.

MySQL Workbench

File Edit View Database Tools Scripting Help

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

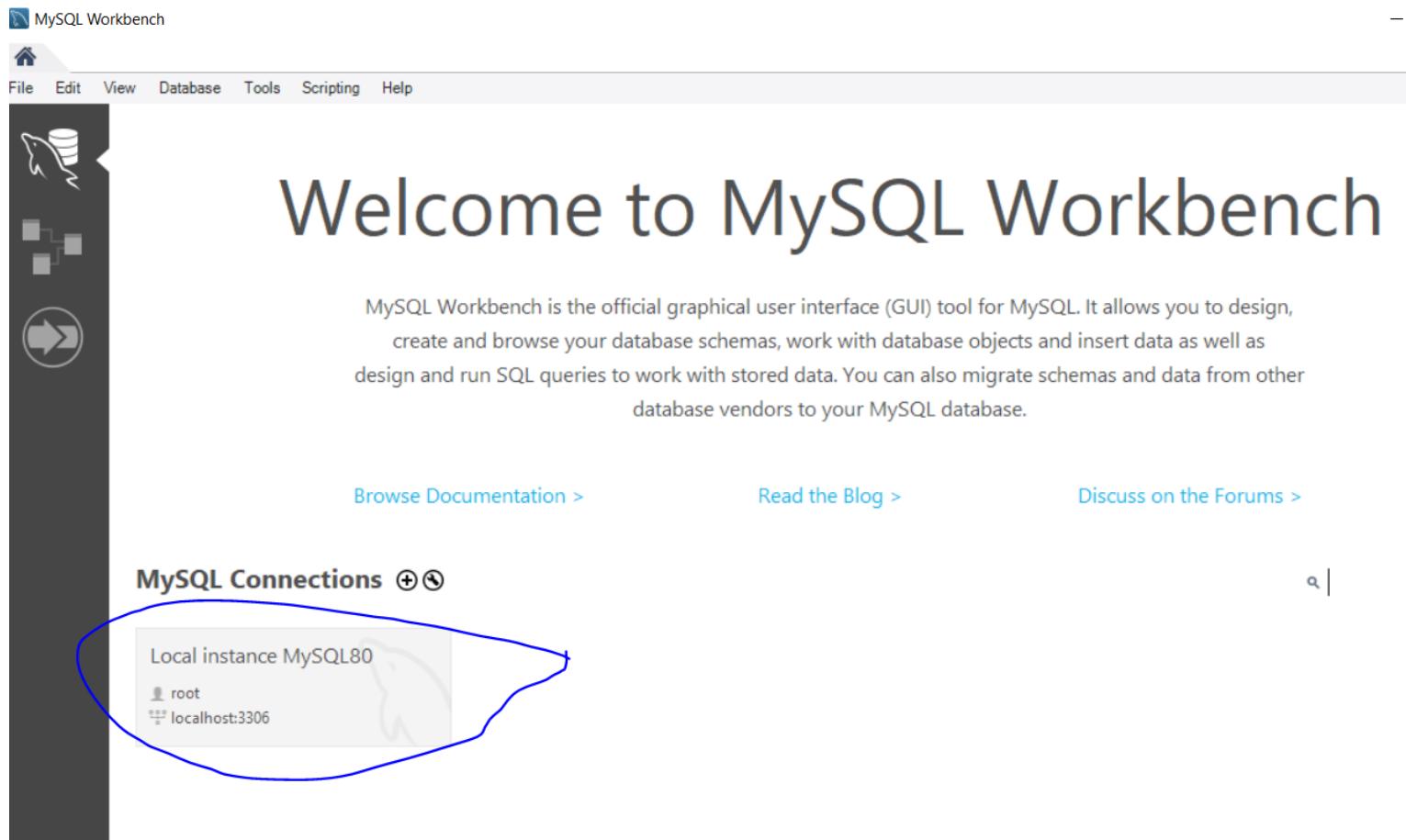
Browse Documentation > Read the Blog > Discuss on the Forums >

MySQL Connections ⊕ ⊗

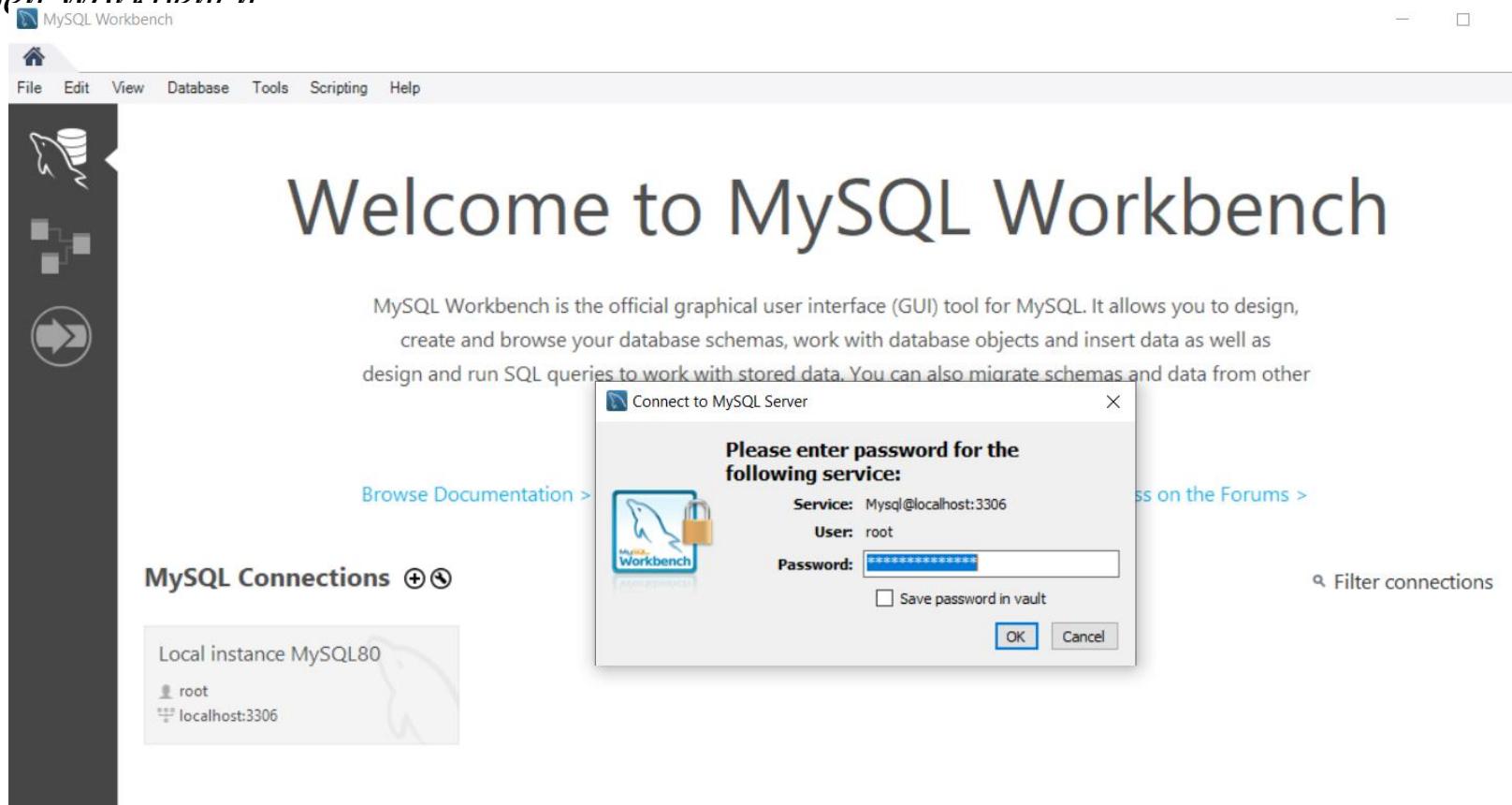
Local instance MySQL80

root
localhost:3306

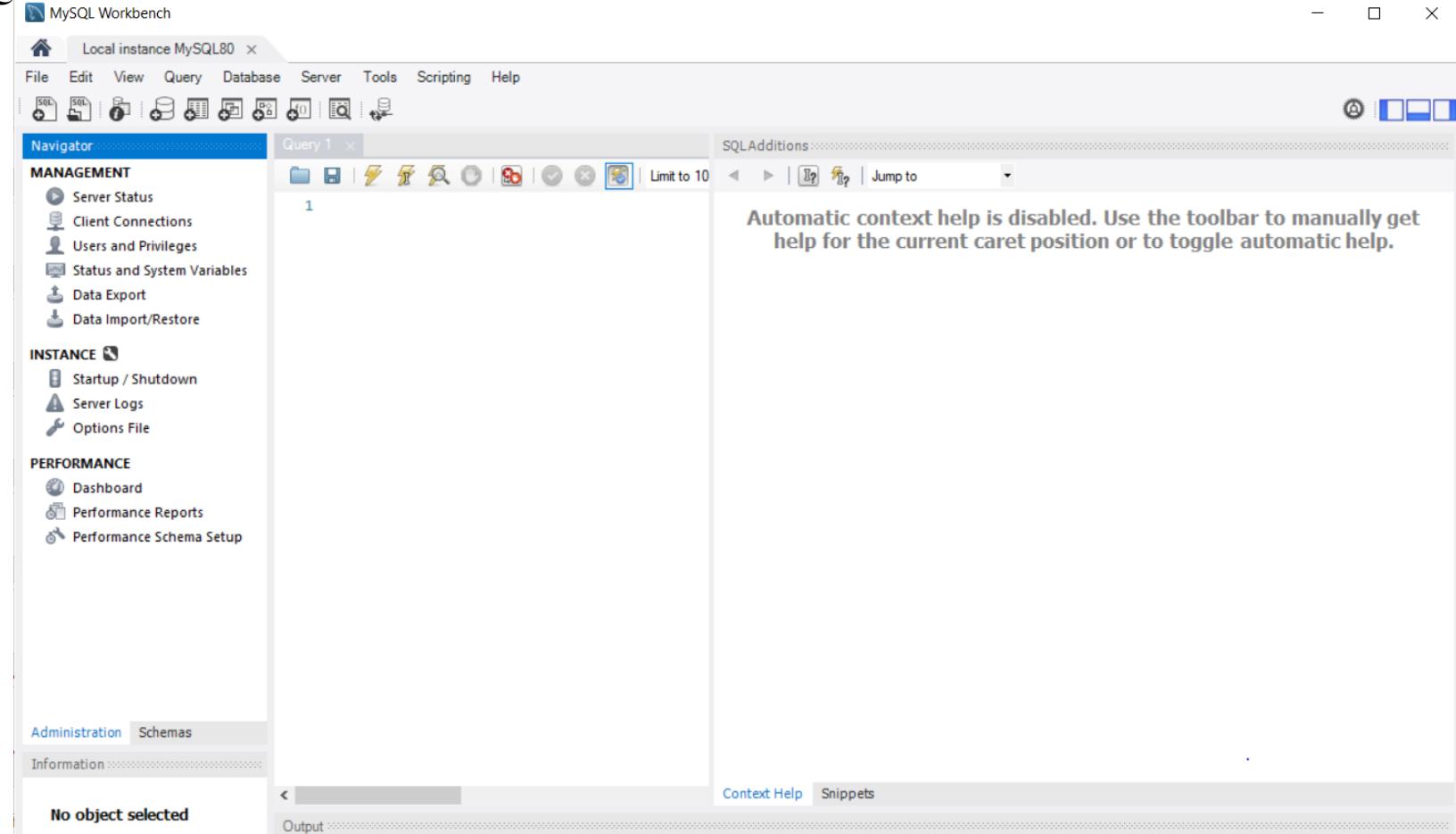
Click on Local instance MySQL8.0 to open Workbench.



Click on Local instance MySQL8.0 to login with root credentials and open Workbench



MySQL8.0 Workbench opens.



MySQL8.0 Workbench opens.

