DEMO NOTES FOR SIMPLE WEB APPLICATION USING SPRING BOOT IN VISUAL STUDIO CODE (VSCode)

This guide will help you create a basic web application using Spring Boot, a popular framework that simplifies Java development, especially for web applications. The demo covers setting up a project in Visual Studio Code (VSCode), creating a REST API, and running the application.

Step 1: Set Up Your Development Environment

1. **Install Required Extensions:**
   - Make sure you have the **Spring Boot Extension Pack** and **Extension Pack for Java** installed on VSCode.
   - These extensions provide tools and support for developing Spring Boot applications. They offer features like project templates, code completion, and debugging support for Java development.

Step 2: Create a New Spring Boot Project Using Spring Initializr

1. **Open the Command Palette:**
   - Press `Ctrl+Shift+P` to open the **Command Palette** in VSCode. The Command Palette is a quick way to access various functionalities within the editor.
2. **Search for Spring Initializr:**
   - Type **"Spring Initializr"** in the search bar and select **"Spring Initializr: Create a Maven Project"**.
   - Spring Initializr is a web-based tool that helps you bootstrap a new Spring Boot project by providing options to choose the project dependencies, packaging type, and other settings.
3. **Follow the Prompts:**
   - **Select Spring Boot Version:** Choose `3.3.3`. This is the latest stable version of Spring Boot. Make sure you select a version that is compatible with your Java Development Kit (JDK).
   - **Choose Project Language:** Select `Java`. Spring Boot supports multiple languages, but Java is the most commonly used.
   - **Input Group ID:** Enter `"com.cs3300"`. The Group ID typically represents the base package name of the project and follows a reverse domain name pattern.
   - **Input Artifact ID:** Enter `"spring_demo"`. The Artifact ID is the name of the project and is also used to name the output artifact (JAR/WAR file).
   - **Specify Packaging Type:** Choose `Jar`. JAR packaging will create a standalone executable JAR file with an embedded server, suitable for running the application independently.
   - **Select Java Version:** Choose `17`, `21`, or `22`. Make sure this version matches the JDK installed on your machine.
   - **Add Dependencies:** Select **Spring Web**. Since we are building a web application, this dependency will include the necessary libraries to create REST APIs, handle HTTP requests, etc.
4. **Confirm Dependencies:**

- After selecting the dependencies, ensure that **Spring Web** is listed, and press **Enter** to continue. This step is crucial as missing dependencies could cause your project not to function as expected.

5. **Choose Project Location:**
   - Choose a location on your computer where you want to save your new project. After selecting the location, click **Open** to open the project in VSCode.
6. **Verify the Project Setup:**
   - Open the `pom.xml` file. This file is the Maven configuration file for your project. It should contain the `spring-boot-starter-web` dependency:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

   - The presence of this dependency ensures that all necessary libraries for creating a web application are included.

Step 3: Create a REST API Endpoint

1. **Define a REST Controller:**
   - Navigate to `src/main/java/com/cs3300/spring_demo` in your project directory.
   - Create a new Java class named `StudentsController.java`.
2. **Annotate the Class as a REST Controller:**
   - Inside `StudentsController.java`, use the `@RestController` annotation to define the class as a REST controller. This tells Spring Boot that the class will handle HTTP requests:

```
// Import necessary packages
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
import java.util.List;
import java.util.Arrays;

// Define the class as a REST controller
@RestController
public class StudentsController {

    // Create a GET endpoint to return a list of students
    @GetMapping("/students")
    public List<Student> getAllStudents() {
        // Return a list of students as JSON response
        return Arrays.asList(
            new Student(1, "Michael"),
            new Student(2, "Alice")
        );
    }
}
```

   - **Explanation**:

- `@RestController` marks the class as a controller where every method returns a domain object instead of a view.
- `@GetMapping("/students")` defines a GET endpoint that listens on the `/students` URL path. When accessed, it returns a list of students.

3. **Create the `Student` Class:**
   - Inside the same package, create a new Java class named `Student.java`. This class represents the data model for the students.
   - Define the following fields and methods:

```
public class Student {
    // Fields
    private int id;
    private String name;

    // Constructor to initialize fields
    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    // Getter and Setter for ID
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    // Getter and Setter for Name
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    // toString method to print details
    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }
}
```

   - **Explanation**:
     - This `Student` class has two fields: `id` and `name`.
     - It includes a constructor to initialize these fields, getters, and setters for accessing and modifying them, and a `toString` method for output.

Step 4: Run the Application

1. **Run the Application:**
   - In VSCode, click on the **Run** button, or open the integrated terminal and run:

```
./mvnw spring-boot:run
```

   - Alternatively, you can build the JAR file and run it using:

```
mvn clean package
java -jar target/spring_demo-0.0.1-SNAPSHOT.jar
```

   - This will start the embedded server (like Tomcat), and your application will be up and running.
2. **Check for Port Conflicts:**
   - If port `8080` is already occupied by another process, change the port by adding the following line to `src/main/resources/application.properties`:

```
server.port = 8081
```

   - This configures the application to run on port `8081` instead.
3. **Access the REST API:**
   - Open your web browser and navigate to `http://localhost:8080/students` (or `http://localhost:8081/students` if you changed the port).
   - You should see a JSON response with the list of students:

```
[
  {"id": 1, "name": "Michael"},
  {"id": 2, "name": "Alice"}
]
```

Step 5: Debugging and Configuration

1. **Enable Debugging:**
   - To see more detailed logs for debugging purposes, add the following line to your `application.properties` file:

```
logging.level.org.springframework=DEBUG
```

   - This will provide additional information about what is happening during the startup and execution of the application, making it easier to identify any issues.

Conclusion:

By following these steps, you have successfully created a simple Spring Boot web application with a REST API endpoint that returns a list of students. This demonstration shows how easy it is to get started with Spring Boot using VSCode, from setting up the project to running a functional web service. You can further expand this application by adding more endpoints, services, and integrating a database.