Search...

How to Create a Simple Spring Application?

Last Updated : 23 Jul, 2025

Creating a Spring application is a good way to get started with this framework. It helps you understand the core principles of Spring, like Inversion of Control (IoC) and Dependency Injection (DI), which promote loosely coupled and easily testable code.

In this article, we explored how to set up and create a Spring application on our system.

Prerequisites:

- Install Java Development Kit (JDK).
- Download and install IntelliJ.
- Understanding of core Java concepts like multithreading, exception handling or the collection framework.

Step-by-Step Implementation

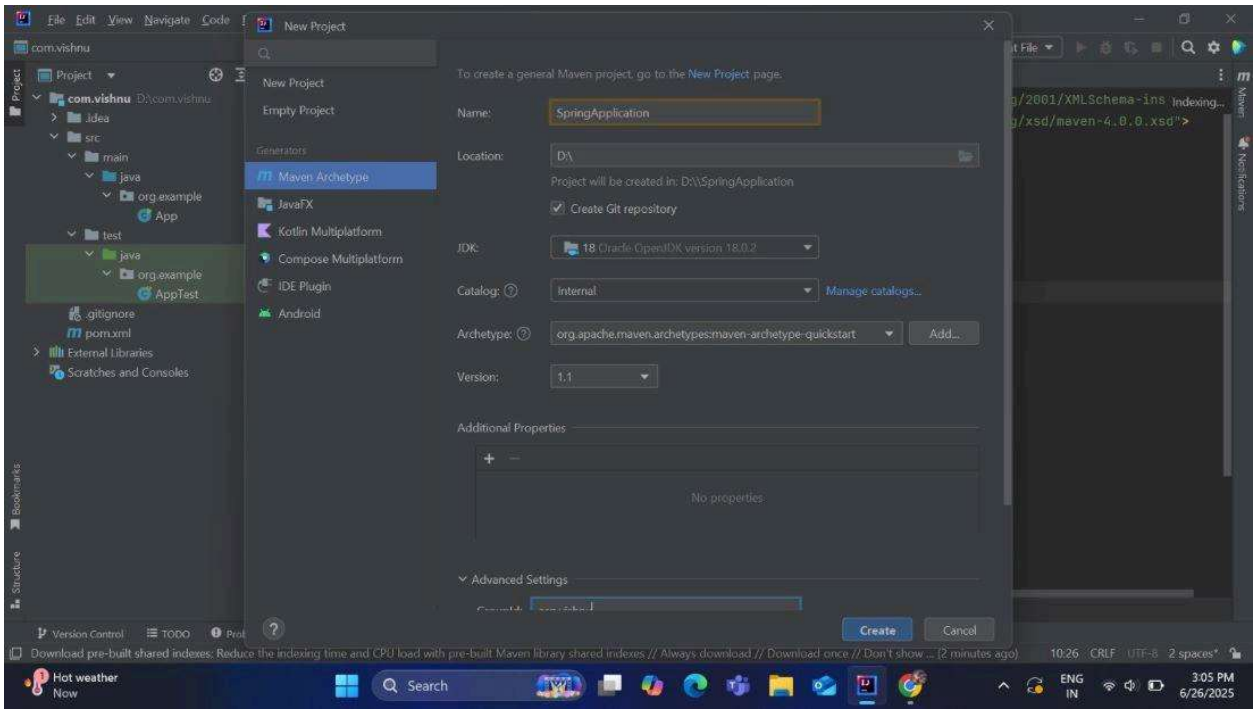
Step 1: Create a Maven project

Let's start by creating a simple **Maven-based Spring** project following step by step.

- Open Intellij IDEA.
- Navigate to the File **tab->Click new->Project.**
- Choose Maven as a project type.
- Click Next and select Quick Start Archetype.
- Fill in the details like.
  - **GroupId:** com.Vishnu,
  - **ArtifactId:** SpringApplication.

- Choose a location for the project and click Finish.

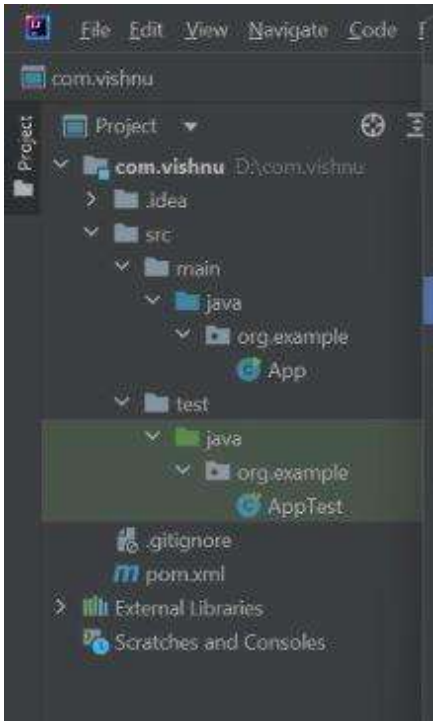
The image below describes the setup of the project:



step1

Step 2: Project Structure

After creating the project successfully and adding the necessary dependencies to your Spring application. The project structure will looks something like this.



step2

Step 3: Add the Necessary Dependency to the pom.xml File.

Maven allows us to manage dependencies automatically, we don't have to download JAR files manually. Maven does it automatically. When we add dependencies in the pom.xml, Maven downloads the important libraries from the central repository and keeps them up to date.

Follow these steps below to add dependencies to pom.xml:

- Open the pom.xml file.
- Visit the Maven repository site to add necessary dependencies.
- Add dependencies inside pom.xml.
- And right click reload maven dependencies.

pom.xml:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
         https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>spring-demo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
```

```
    <!-- Spring Core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.3.29</version>
    </dependency>

    <!-- Spring Context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.29</version>
    </dependency>
</dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Step 4: Open the Main class and Run

If everything is set up correctly, then your spring application will run successfully, and you will see the desired output on the console.

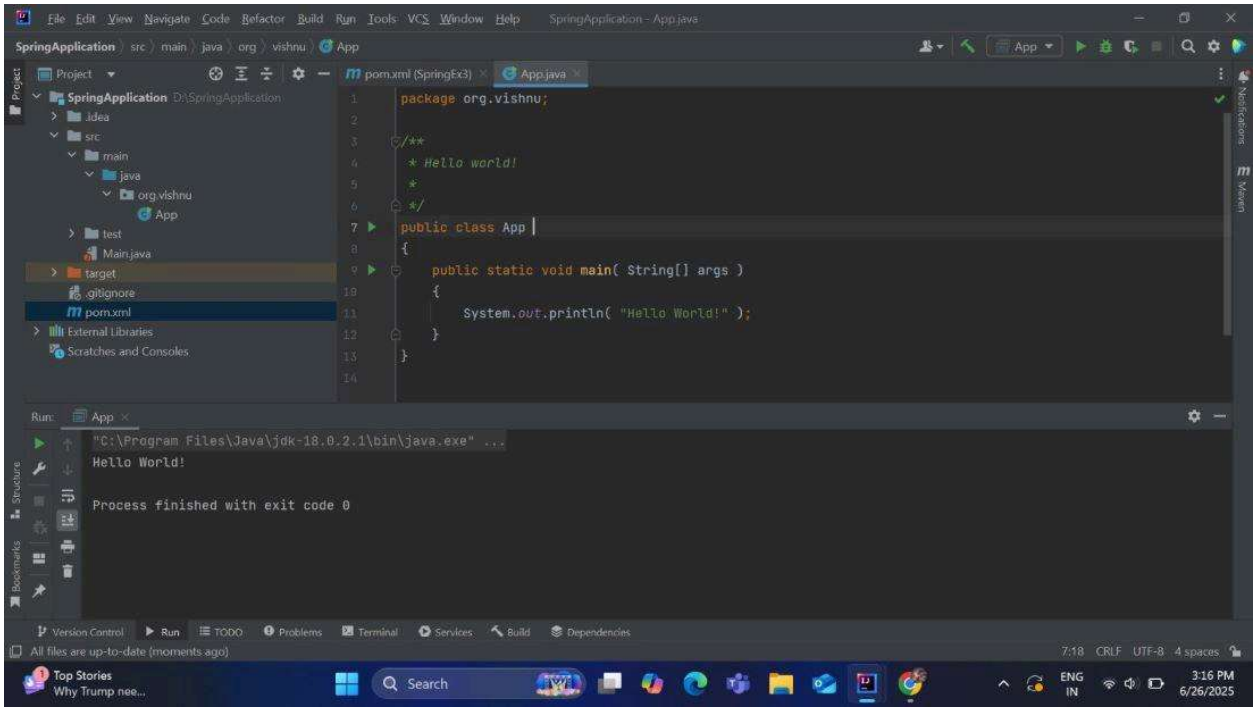
App.java:

```
package org.vishnu;

public class App {

    public static void main(String[] args)
    {
        System.out.println("Hello World.");
    }
}
```

Output:



step4

Comment More info

Campus Training Program