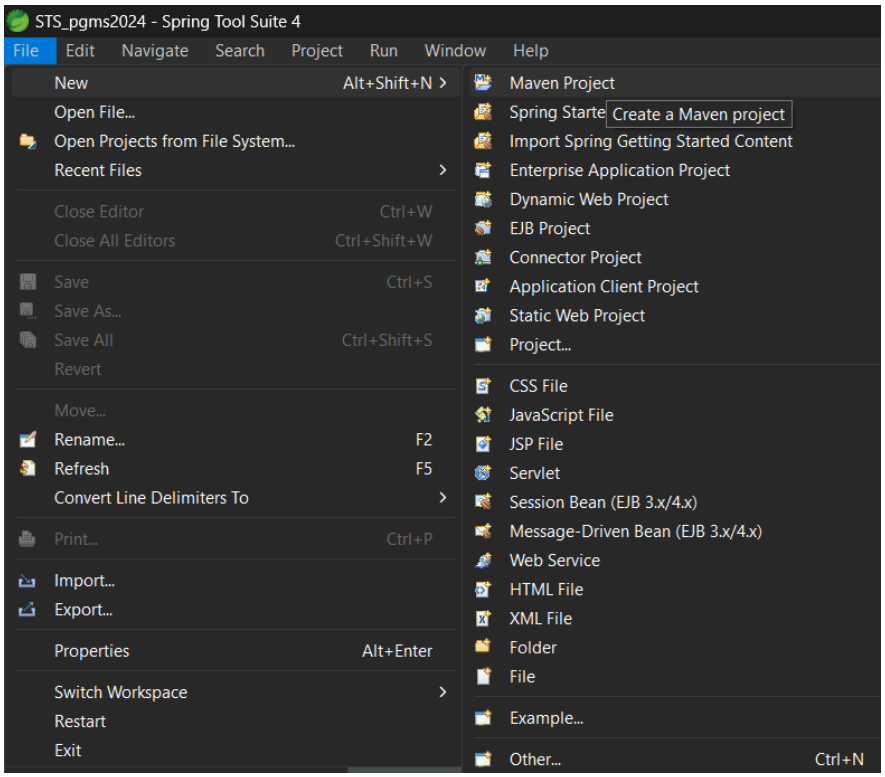
Spring without Boot

Thus far, we have observed the remarkable attributes of Spring Boot and how it simplifies the lives of developers, allowing them ample time to focus on business logic. However, what exactly is occurring behind the scenes? We require an understanding of how spring operates. Now, let's delve into the functionality of the Spring framework.

Spring without Boot:
While working with frameworks, we have also worked with popular project management build tools like ***Maven*** and ***Gradle.***
Maven is one of the most popular build automation tools. It is developed using the Java programming language and is primarily used for Java-based projects.
Here we are going to work with the Maven build tool.

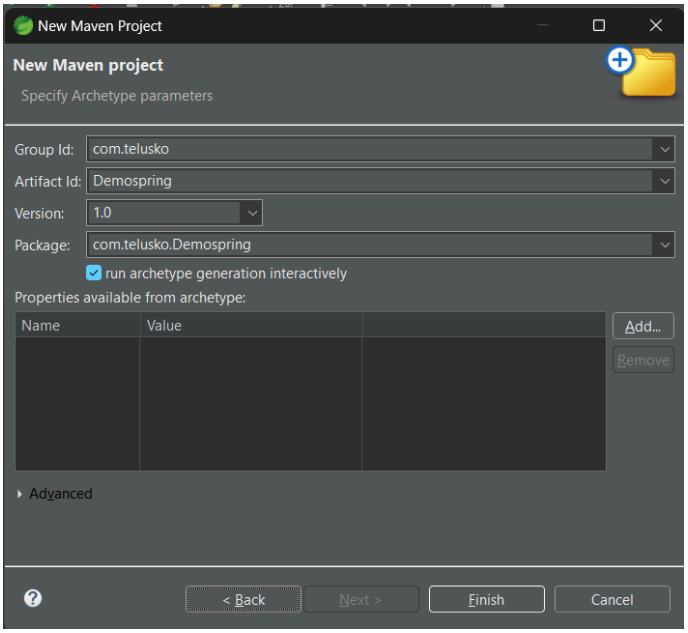
Steps to create a maven project:

Step 1:
First, open your Spring Tool Suite.
After that, click on the file menu option.
Then select the new option, and then select Maven Project.

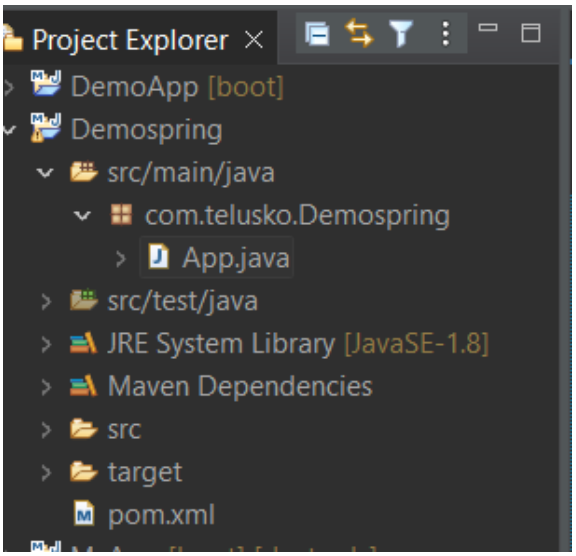


Step 2:

A new window has now been opened. We must enter project details in this window, such as versions, language, package name, artifact ID, and other data. Project type in this case is Maven, and package and artifact ID are provided.



Step 3:



After the necessary project internal dependencies have been downloaded to the local machine, Maven will generate a standard project folder structure. Here is an example of a project folder structure.

Since we are currently developing projects with the Spring framework, it is necessary to include the Spring dependency in the pom.xml file.

To obtain the necessary dependency, simply launch any web browser and navigate to the following website**:**
https://mvnrepository.com/artifact/org.springframework/spring-context.
Select the most recent version that has the fewest vulnerabilities. I recommend choosing version 6.1.6 and simply copying and pasting it into your project's pom file. Afterward, reload or update your maven project.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xsi:schemaLocation="http://maven.apache.org/POM/
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.telusko</groupId>
  <artifactId>Demospring</artifactId>
  <version>1.0 </version>
  <packaging>jar</packaging>

  <name>Demospring</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.b
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.sp
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>6.1.6</version>
</dependency>

    <dependency>
  </dependencies>
</project>
```

Now let’s delve into code:

Instantiate a class, To demonstrate dependency injection, we will create a method called "build" in the "dev" class. This method will be invoked in the main class. Instead of using the traditional approach of creating a "dev" object using the "new" keyword, we will utilise the

App.java | Demospring/pom.xml | Dev.java | Laptop.java

```java
package com.telusko.Demospring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main( String[] args )
    {

        ApplicationContext context =new ClassPathXmlApplicationContext();
        Dev obj=context.getBean(Dev.class);
        obj.build();
    }
}

```

Spring framework. We will create a container using the ApplicationContext, which we have already learned about.
Since ApplicationContext is an interface, it cannot be instantiated. However, we will be

using its implementing class is ClasspathXmlApplicationContext.
There are multiple methods for creating containers, such as utilizing pure Java-based configuration, XML-based configuration, or annotation-based configuration.
XML is being used in our example.
By utilizing this context, we can obtain the desired instance of the dev class, which is currently stored in the container. However, in order to proceed, it is necessary to configure the container. Since we are utilizing an XML-based configuration, an XML file is required.

The upcoming chapter will cover XML configuration, including its syntaxes, writing style, and background operations.
.