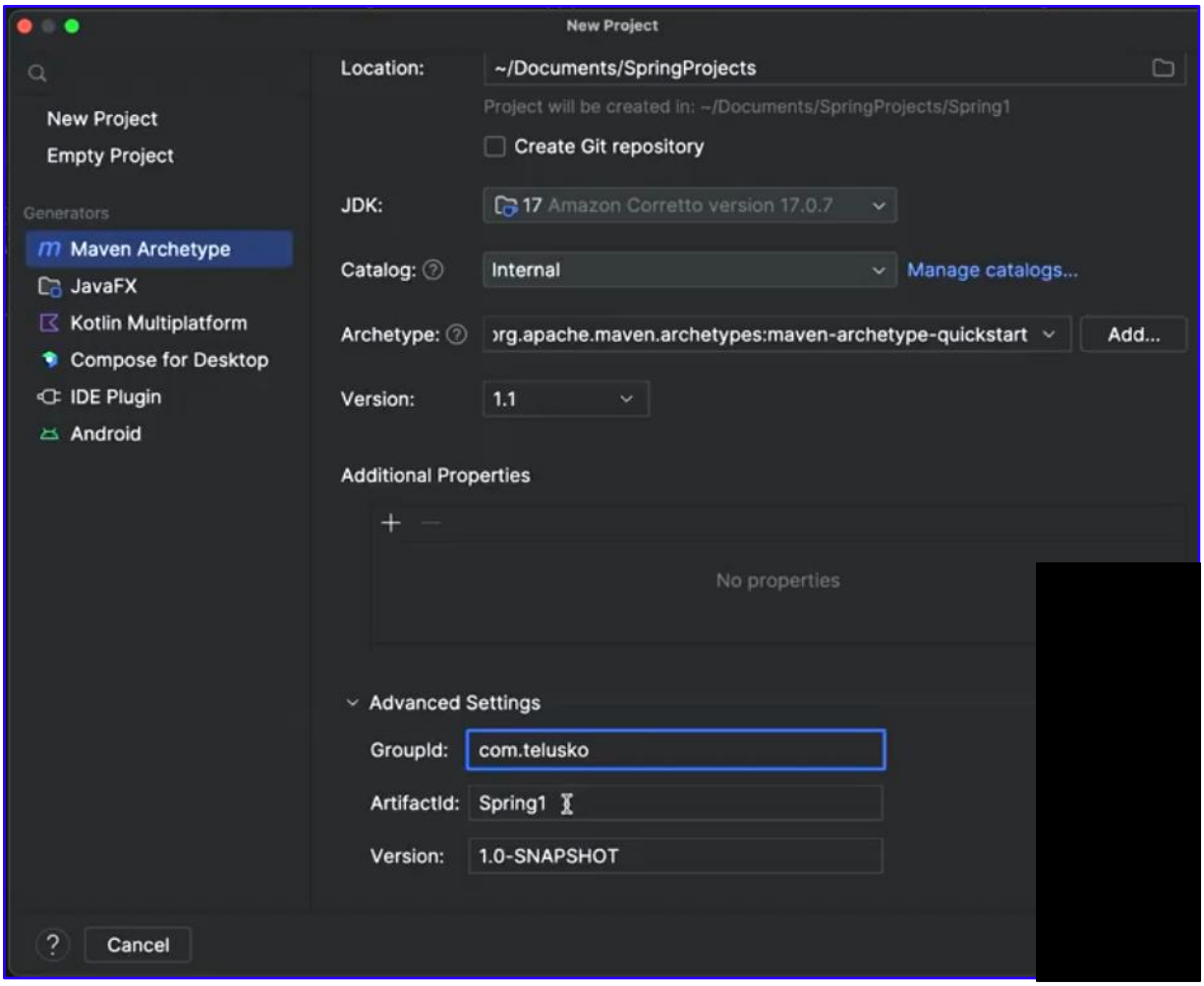
01 - SPRING 1st PROJECT

👉 Steps to Setup a Maven Spring Project:

Create a new **Maven project** using your IDE or CLI.

- **Project Name:** Assign a suitable name to the project (e.g., SpringDemo).
- **Location:** Choose a directory to store your project.
- **JDK Version:** Ensure **JDK 17** or higher is selected (required for Spring 6 framework).
- **Catalog:** Select **Internal** as the catalog.
- **Archetype:** Choose **QuickStart** to create the basic project structure.
- **Version:** Set it to **1.1**.
- **GroupId:** Represents the project's root package (e.g., com.telusko).
- **ArtifactId:** The project's name (e.g., spring-app).
- **Version:** Usually set to **1.0-SNAPSHOT**.



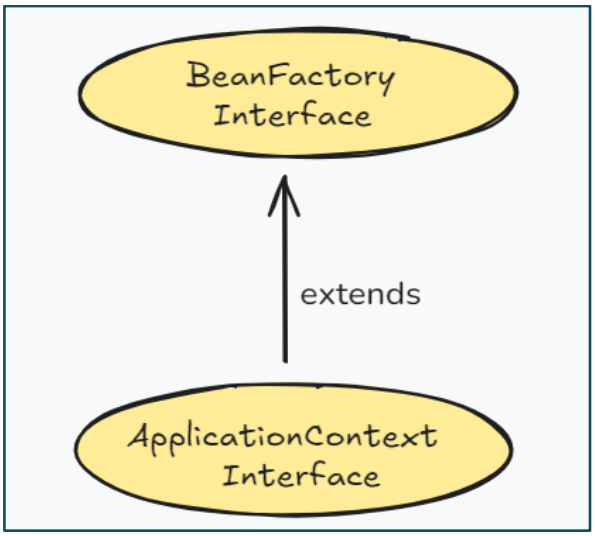
➢ Once the Maven project is created, the pom.xml file should include basic dependencies and configurations.

➢ Verify the project structure by running the App.java file. You should see "Build Success" if the project is correctly set up.

👉 IOC (Inversion of Control) Container:

- The IOC Container in Spring is responsible for managing the lifecycle of objects (called beans), their configurations, and their dependencies.
- It inverts the control of object creation from the developer to the framework.
- IOC container also injects the necessary dependencies.
- The two key types of IOC containers in Spring are:
    - BeanFactory: A basic container for managing beans.

- ApplicationContext: A more feature-rich container that builds upon BeanFactory.



👉 BeanFactory Interface:

- ➢ BeanFactory is a lightweight container in Spring that provides basic DI and bean management functionalities.
- ➢ It follows the l**azy loading principle**, meaning beans are only created when requested

👉 ApplicationContext Interface:

ApplicationContext is an advanced IOC container in Spring. It builds on top of BeanFactory and provides additional features like:

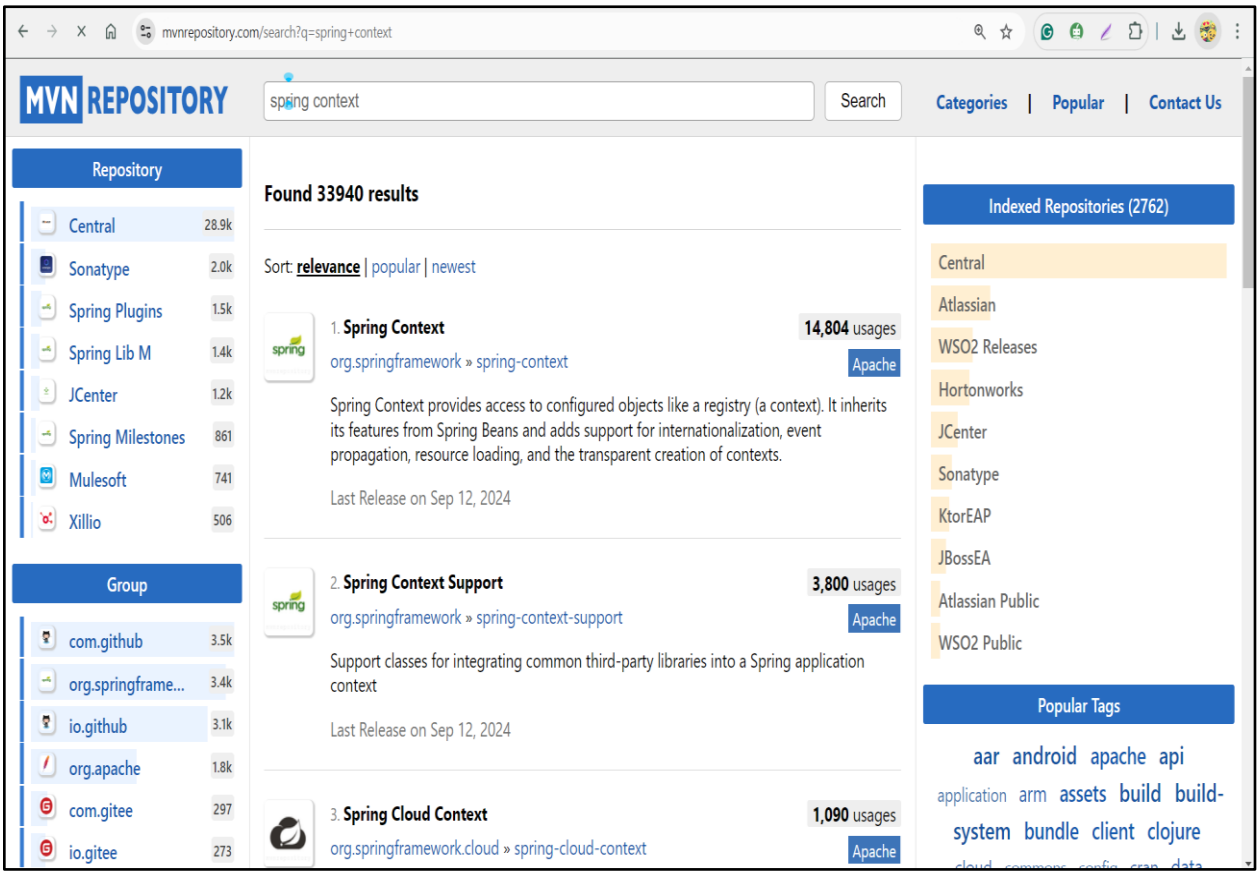
- ➢ **Event propagation**: Handles events published by beans.
- ➢ **AOP (Aspect-Oriented Programming)**: Supports declarative AOP features.
- ➢ **Message resource handling**: Allows support for internationalization and easy access to resource bundles.
- ➢ **Bean Autowiring**: Automatic injection of dependencies based on bean types.
- ➢ **Different Context Types**: Provides several specialized types of ApplicationContext implementations, such as:
  - **ClassPathXmlApplicationContext**: Loads context from an XML file located in the classpath.

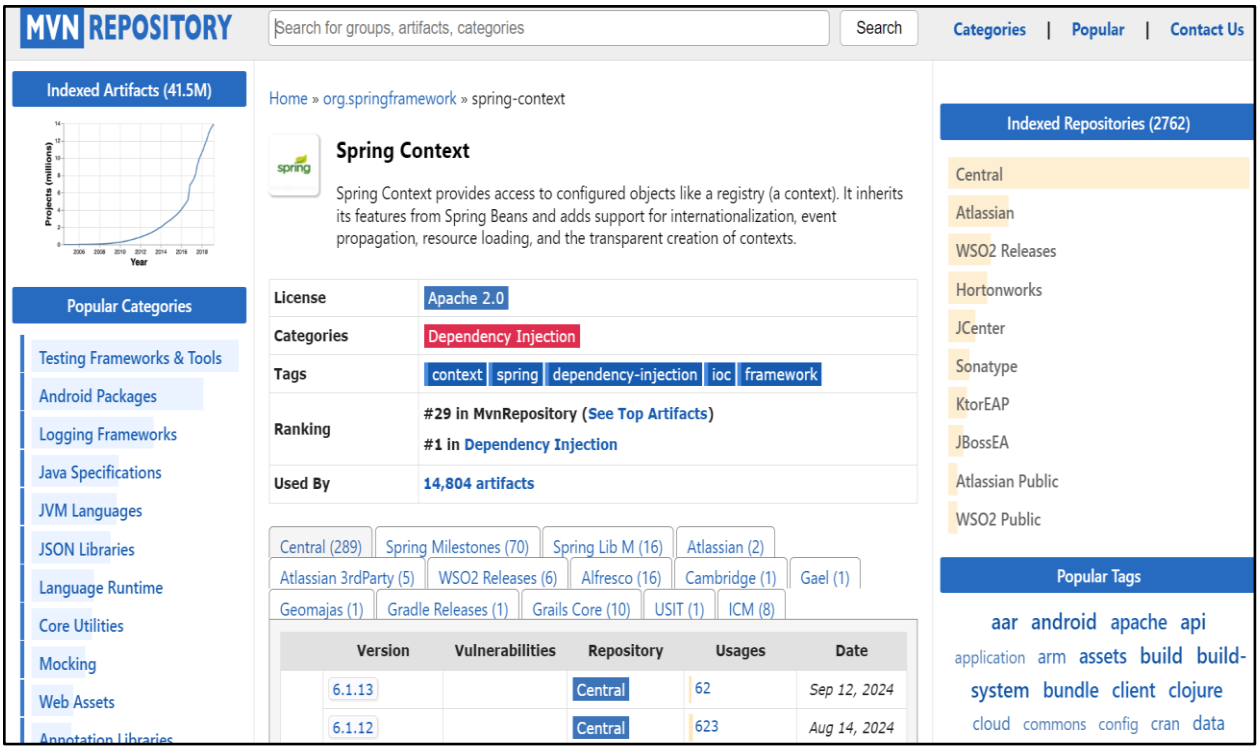
- **FileSystemXmlApplicationContext**: Loads context from an XML file outside the classpath.
- **AnnotationConfigApplicationContext**: Supports Java-based configuration without XML.

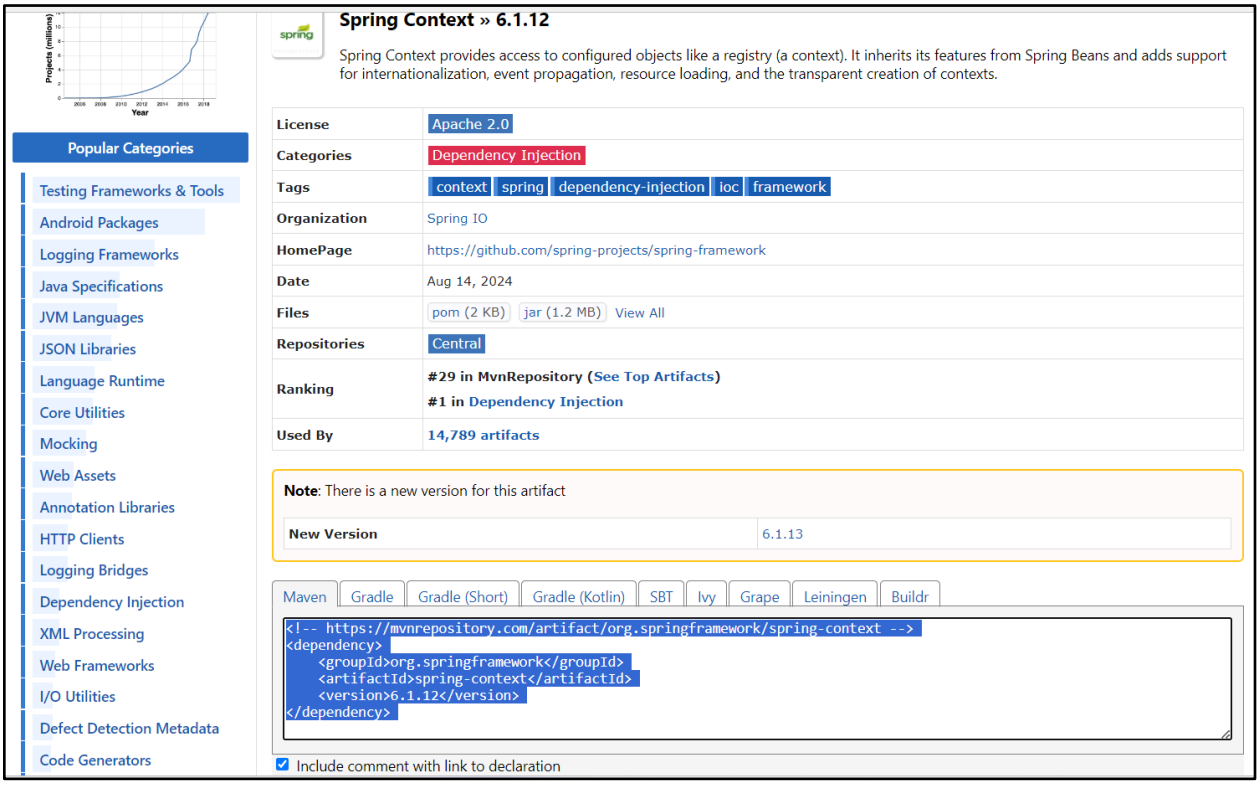
👉 Adding Spring Context Dependency:

To use the ApplicationContext, add the **spring-context** dependency in pom.xml

Website Link: https://mvnrepository.com/







Dependency:

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>6.1.12</version>
</dependency>
```

👉 XML-Based Configuration with ClassPathXmlApplicationContext:

You need to use ClassPathXmlApplicationContext to load bean definitions from an XML configuration file.

Example code to create a Spring container:

```
ApplicationContext context = new ClassPathXmlApplicationContext();
  Alien obj = (Alien)context.getBean("alien");
  obj.code();
```

- ApplicationContext -> manages the lifecycle and configuration of Spring beans.
- ClassPathXmlApplicationContext -> loads the bean configuration from an XML file located on the classpath
- getBean() -> method fetches a bean with the specified id (in this case, "alien") from the IOC container. It returns an object of type Object, so explicit casting is required to convert it to the type of the desired bean
- Alien obj = (Alien) -> ensures that the returned bean is treated as an object of the Alien class
- obj.code() -> calls the code() method on the Alien object

👉 Common Error: BeanFactory Not Initialized

The error occurs when accessing a bean before the Spring context is properly initialized. The Spring container (ApplicationContext) is not fully loaded or closed.

Error:

Exception in thread "main" java.lang.IllegalStateException: BeanFactory not initialized or already closed - call 'refresh' before accessing beans via the ApplicationContext at org.springframework.context.support.AbstractRefreshableApplicationContext.getBeanFactory(AbstractRefreshableApplicationContext.java:169)at org.springframework.context.support.AbstractApplicationContext.getBean(AbstractApplicationContext.java:1243)at com.telusko.App.main(App.java:18)

Code Link:

https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/2%20Exploring%20Spring%20Framework/2.1%20Spring%201st%20Project/Spring1