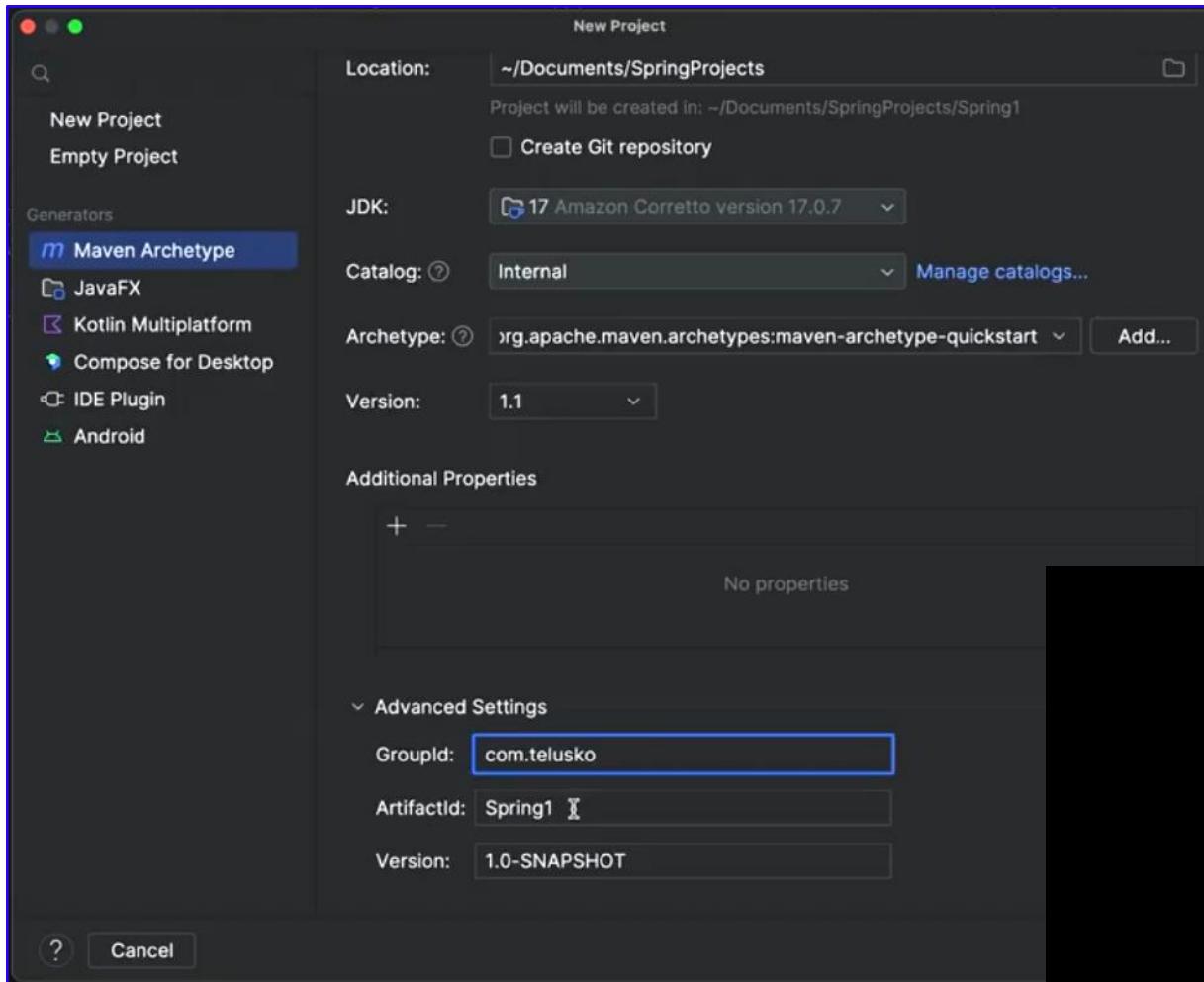


01 - SPRING 1st PROJECT

👉 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**.

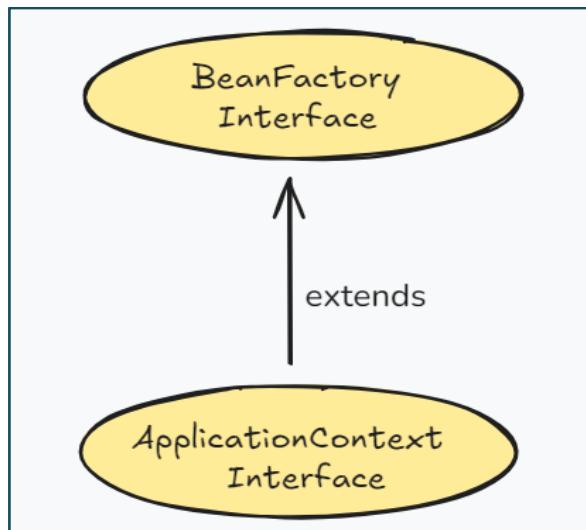


- Once the Maven project is created, the `pom.xml` file should include basic dependencies and configurations.
- Verify the project structure by running the `App.java` file. You should see "Build Success" if the project is correctly set up.

👉 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 **lazy 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/>

The screenshot shows the MVN Repository website interface. The search bar at the top contains the query "spring context". Below the search bar, the main content area displays the results for "Found 33940 results". The first result is "1. Spring Context" by org.springframework, which has 14,804 usages and is Apache-licensed. It provides a brief description of Spring Context and its last release date. The second result is "2. Spring Context Support" by org.springframework, with 3,800 usages and Apache licensing. The third result is "3. Spring Cloud Context" by org.springframework.cloud, with 1,090 usages and Apache licensing. On the left side, there are two navigation panels: "Repository" and "Group". The "Repository" panel lists various repositories like Central, Sonatype, and JCenter. The "Group" panel lists groups like com.github, org.springframework, and io.github. On the right side, there are two sidebar panels: "Indexed Repositories (2762)" and "Popular Tags". The "Indexed Repositories" sidebar lists several repositories, with Central being the most prominent. The "Popular Tags" sidebar lists various tags such as aar, android, apache, api, application, arm, assets, build, build-system, bundle, client, clojure, cloud, common, config, cran, data, etc.

MVN REPOSITORY

Search for groups, artifacts, categories

Categories | Popular | Contact Us

Indexed Artifacts (41.5M)

Project (millions) vs Year

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries

Spring Context

Spring Context provides access to configured objects like a registry (a context). It inherits its features from Spring Beans and adds support for internationalization, event propagation, resource loading, and the transparent creation of contexts.

License	Apache 2.0
Categories	Dependency Injection
Tags	context spring dependency-injection ioc framework
Ranking	#29 in MvnRepository (See Top Artifacts) #1 in Dependency Injection
Used By	14,804 artifacts

Central (289) Spring Milestones (70) Spring Lib M (16) Atlassian (2)
Atlassian 3rdParty (5) WSO2 Releases (6) Alfresco (16) Cambridge (1) Gael (1)
Geomajas (1) Gradle Releases (1) Grails Core (10) USIT (1) ICM (8)

Version	Vulnerabilities	Repository	Usages	Date
6.1.13		Central	62	Sep 12, 2024
6.1.12		Central	623	Aug 14, 2024

Indexed Repositories (2762)

- Central
- Atlassian
- WSO2 Releases
- Hortonworks
- JCenter
- Sonatype
- KtorEAP
- JBossEA
- Atlassian Public
- WSO2 Public

Popular Tags

- aar
- android
- apache
- api
- application
- arm
- assets
- build
- build-system
- bundle
- client
- clojure
- cloud
- commons
- config
- cran
- data

Spring Context

Project (millions) vs Year

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- HTTP Clients
- Logging Bridges
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Defect Detection Metadata
- Code Generators

Spring Context > 6.1.12

Spring Context provides access to configured objects like a registry (a context). It inherits its features from Spring Beans and adds support for internationalization, event propagation, resource loading, and the transparent creation of contexts.

License	Apache 2.0
Categories	Dependency Injection
Tags	context spring dependency-injection ioc framework
Organization	Spring IO
HomePage	https://github.com/spring-projects/spring-framework
Date	Aug 14, 2024
Files	pom (2 KB) jar (1.2 MB) View All
Repositories	Central
Ranking	#29 in MvnRepository (See Top Artifacts) #1 in Dependency Injection
Used By	14,789 artifacts

Note: There is a new version for this artifact

New Version	6.1.13
-------------	--------

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

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

Include comment with link to declaration

Dependency:

TELUSKO

```
<!-- 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>