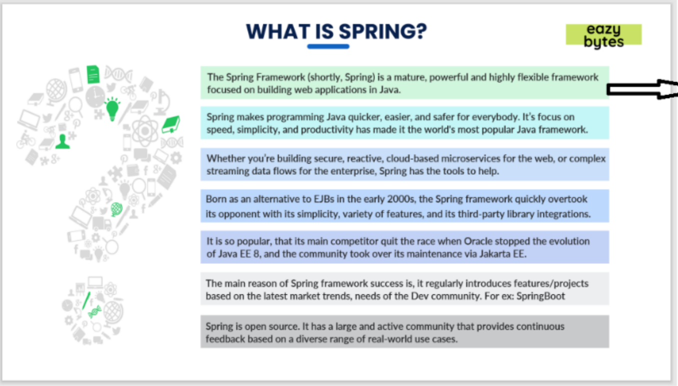


WHAT IS SPRING?

eazy bytes

The Spring Framework (shortly, Spring) is a mature, powerful and highly flexible framework focused on building web applications in Java.

Spring makes programming Java quicker, easier, and safer for everybody. It's focus on speed, simplicity, and productivity has made it the world's most popular Java framework.

Whether you're building secure, reactive, cloud-based microservices for the web, or complex streaming data flows for the enterprise, Spring has the tools to help.

Born as an alternative to EJBs in the early 2000s, the Spring framework quickly overtook its opponent with its simplicity, variety of features, and its third-party library integrations.

It is so popular, that its main competitor quit the race when Oracle stopped the evolution of Java EE 8, and the community took over its maintenance via Jakarta EE.

The main reason of Spring framework success is, it regularly introduces features/projects based on the latest market trends, needs of the Dev community. For ex: SpringBoot

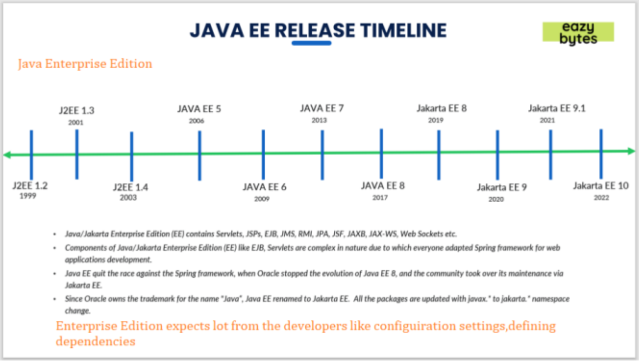
Spring is open source. It has a large and active community that provides continuous feedback based on a diverse range of real-world use cases.

As an developer we should be more focussed on writing the bussiness logic rather than writing dependencies and settuing up configurations.

EJB -Enterprise Java Beans:
EJBS invented as part of Java Enterprise edition.It is complex to develop web applications with EJBs

Spring is Open Source and has large community support as well

Jakarta vs Spring:



JAVA EE RELEASE TIMELINE

eazy bytes

Java Enterprise Edition

- *Java/Jakarta Enterprise Edition (EE) contains Servlets, JSPs, EJB, JMS, RMI, JPA, JSF, JAXB, JAX-WS, Web Sockets etc.*
- *Components of Java/Jakarta Enterprise Edition (EE) like EJB, Servlets are complex in nature due to which everyone adapted Spring framework for web applications development.*
- *Java EE quit the race against the Spring framework, when Oracle stopped the evolution of Java EE 8, and the community took over its maintenance via Jakarta EE.*
- *Since Oracle owns the trademark for the name "Java", Java EE renamed to Jakarta EE. All the packages are updated with javax.* to jakarta.* namespace change.*

Enterprise Edition expects lot from the developers like configuiration settings,defining dependencies

Strandard Edition
Ex: Exception Handling
Collection API

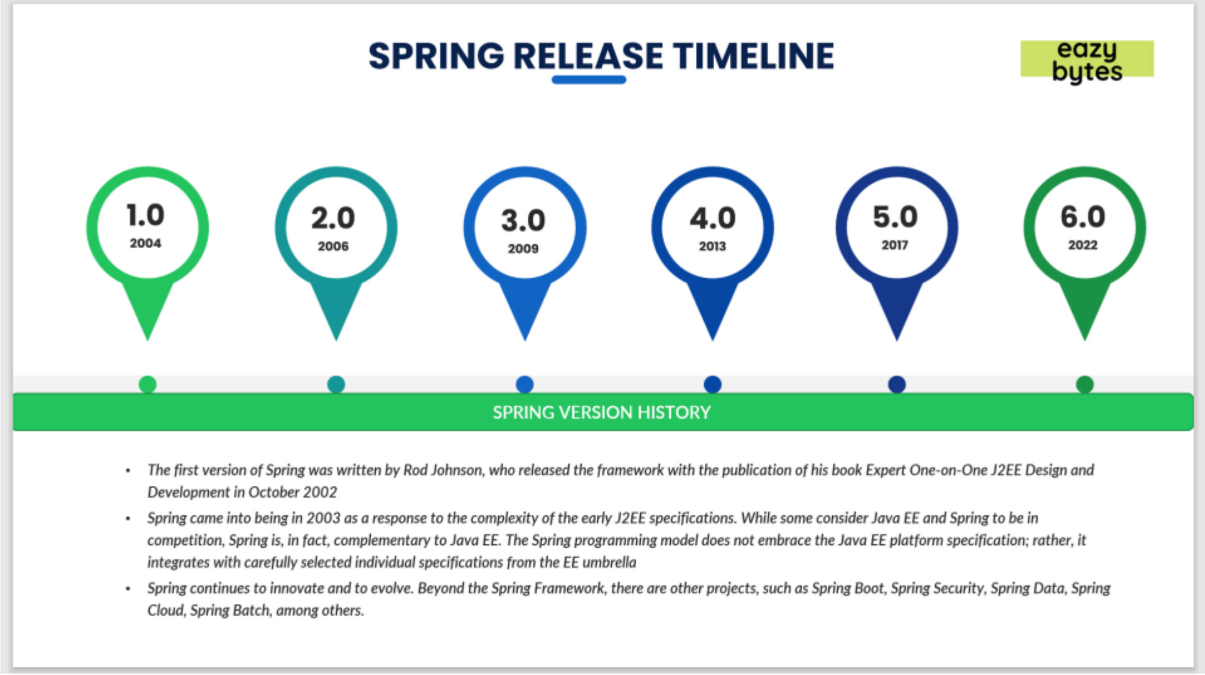
Mobile Edition

Enterprise Edition:
This supports development
web applications

Coniser a scenario where we want to fetch the data from Data base:

IN Java EE:
We should create connection,configirations,Closing the Connections and Exceptions these has to be taken care by the Developer.

Spring:
We have to provide the data base credentails and Spring frame work will take of creating,closing the connection and Exception Handling .We can focus on the bussiness logic .



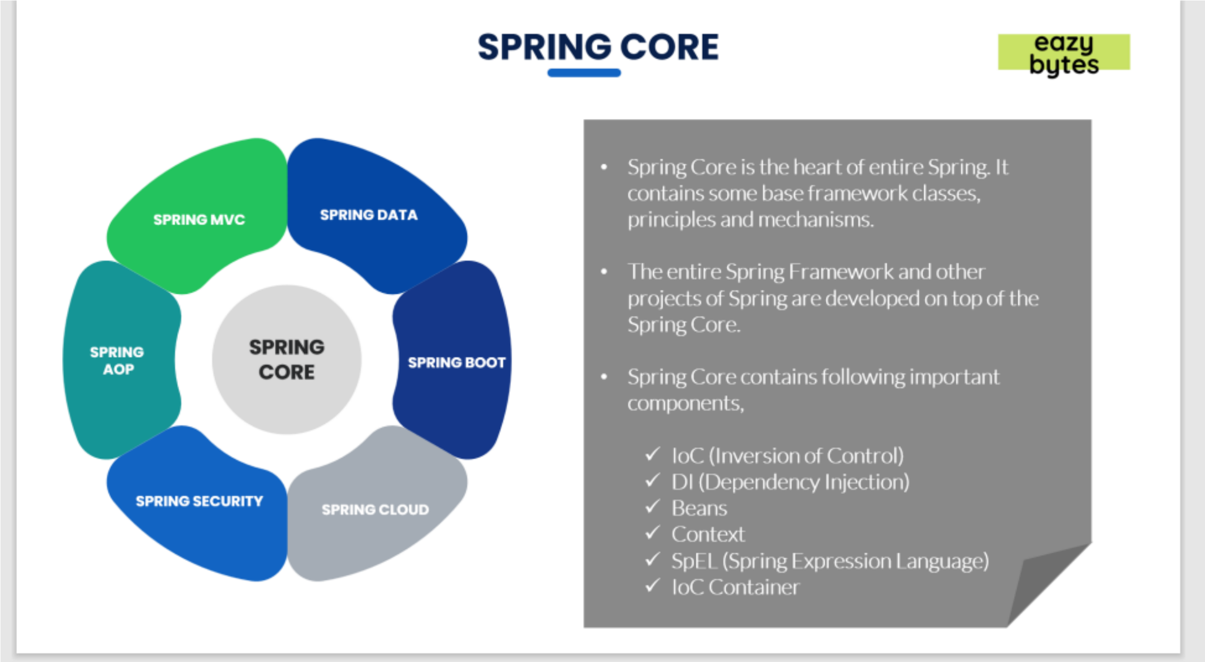
SPRING RELEASE TIMELINE

eazy bytes

SPRING VERSION HISTORY

- *The first version of Spring was written by Rod Johnson, who released the framework with the publication of his book Expert One-on-One J2EE Design and Development in October 2002*
- *Spring came into being in 2003 as a response to the complexity of the early J2EE specifications. While some consider Java EE and Spring to be in competition, Spring is, in fact, complementary to Java EE. The Spring programming model does not embrace the Java EE platform specification; rather, it integrates with carefully selected individual specifications from the EE umbrella*
- *Spring continues to innovate and to evolve. Beyond the Spring Framework, there are other projects, such as Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch, among others.*

Spring Core:

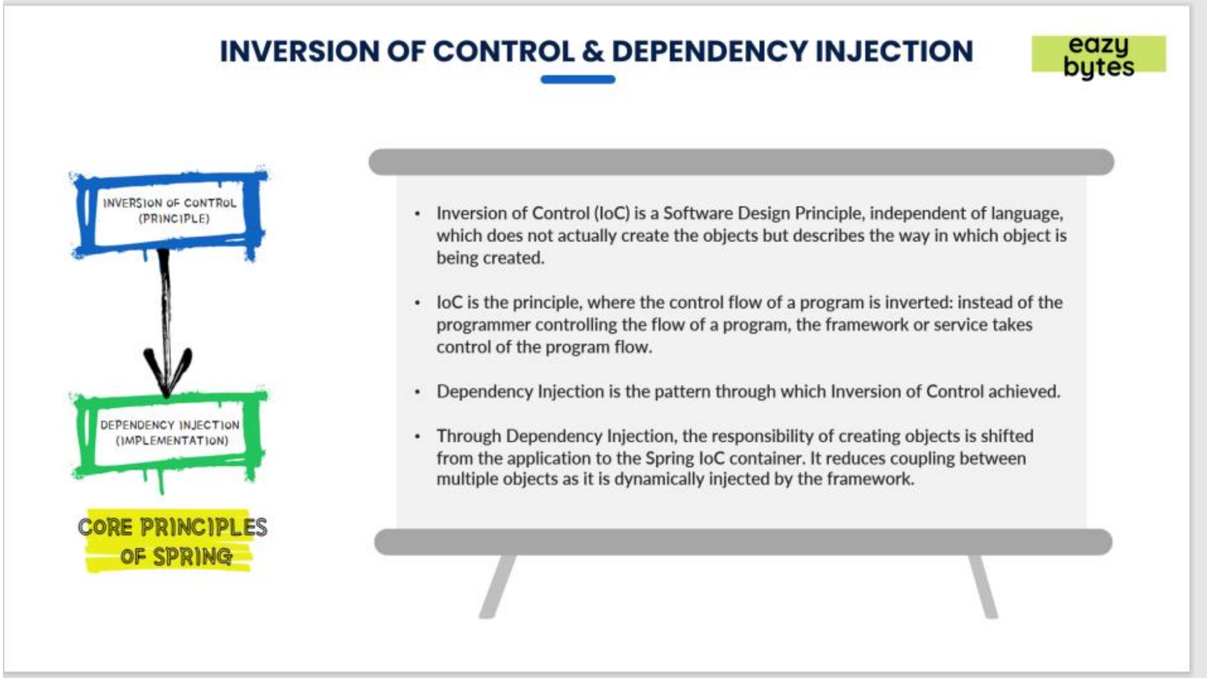


SPRING CORE

- Spring Core is the heart of entire Spring. It contains some base framework classes, principles and mechanisms.
- The entire Spring Framework and other projects of Spring are developed on top of the Spring Core.
- Spring Core contains following important components,
  - ✓ IoC (Inversion of Control)
  - ✓ DI (Dependency Injection)
  - ✓ Beans
  - ✓ Context
  - ✓ SpEL (Spring Expression Language)
  - ✓ IoC Container

Before working with any Spring Framework application we must understand Spring core.

Lecture 6:

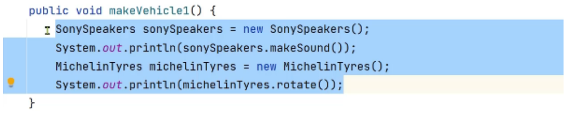


INVERSION OF CONTROL & DEPENDENCY INJECTION

INVERSION OF CONTROL (PRINCIPLE)
DEPENDENCY INJECTION (IMPLEMENTATION)
CORE PRINCIPLES OF SPRING

- Inversion of Control (IoC) is a Software Design Principle, independent of language, which does not actually create the objects but describes the way in which object is being created.
- IoC is the principle, where the control flow of a program is inverted: instead of the programmer controlling the flow of a program, the framework or service takes control of the program flow.
- Dependency Injection is the pattern through which Inversion of Control achieved.
- Through Dependency Injection, the responsibility of creating objects is shifted from the application to the Spring IoC container. It reduces coupling between multiple objects as it is dynamically injected by the framework.

7.Demo of Inversion control and Dependency Injection

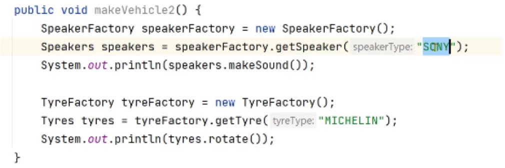


```
public void makeVehicle1() {
    SonySpeakers sonySpeakers = new SonySpeakers();
    System.out.println(sonySpeakers.makeSound());
    MichelinTyres michelinTyres = new MichelinTyres();
    System.out.println(michelinTyres.rotate());
}
```

Coniser you are car company and making vehicle
In this if you could see we are using Sony Speakers and
Some Company Tyres

But at later point of time if we want to change the
speakers to a different company then all the core logic
has to be modified.

Due to there is Tight Coupling between the components.

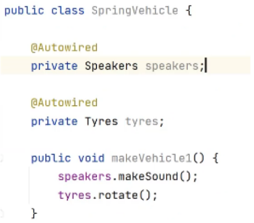


```
public void makeVehicle2() {
    SpeakerFactory speakerFactory = new SpeakerFactory();
    Speakers speakers = speakerFactory.getSpeaker( speakerType: "SONY");
    System.out.println(speakers.makeSound());

    TyreFactory tyreFactory = new TyreFactory();
    Tyres tyres = tyreFactory.getTyre( tyreType: "MICHELIN");
    System.out.println(tyres.rotate());
}
```

For acheiving Inversion control and dependency Injection developers
came up with Factory pattern design pattern but there we need to
pass the Type of speakers/Type of Tyres we need.

If there is any requeiremnt we need to code again.

It is basically like Comapny creates a Speaker Factory and make a
request to get the Speakers of that particualr type.



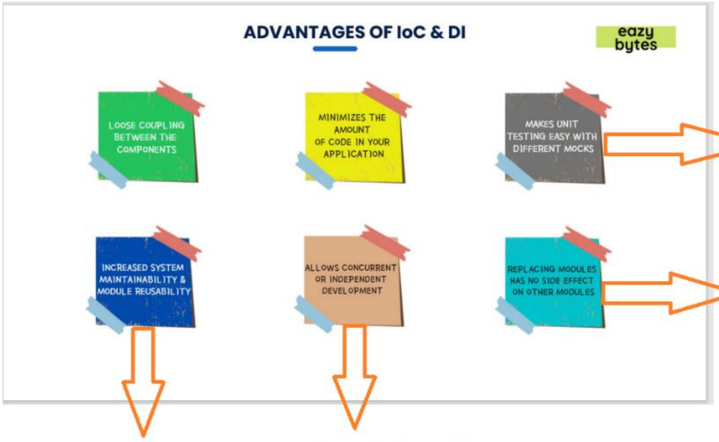
```
public class SpringVehicle {

    @Autowired
    private Speakers speakers;

    @Autowired
    private Tyres tyres;

    public void makeVehicle1() {
        speakers.makeSound();
        tyres.rotate();
    }
```

Based on the Object availablity at the run time corresponding logic will be exceucuted.

Spring Vehicle does not have to be aware of any of which Speaker/Which Tyres.

Here we are just telling Spring Framework we need Speakers and Type for making vehicle.

8.Advantages of Inversion control and Dependency Injection



ADVANTAGES OF IoC & DI

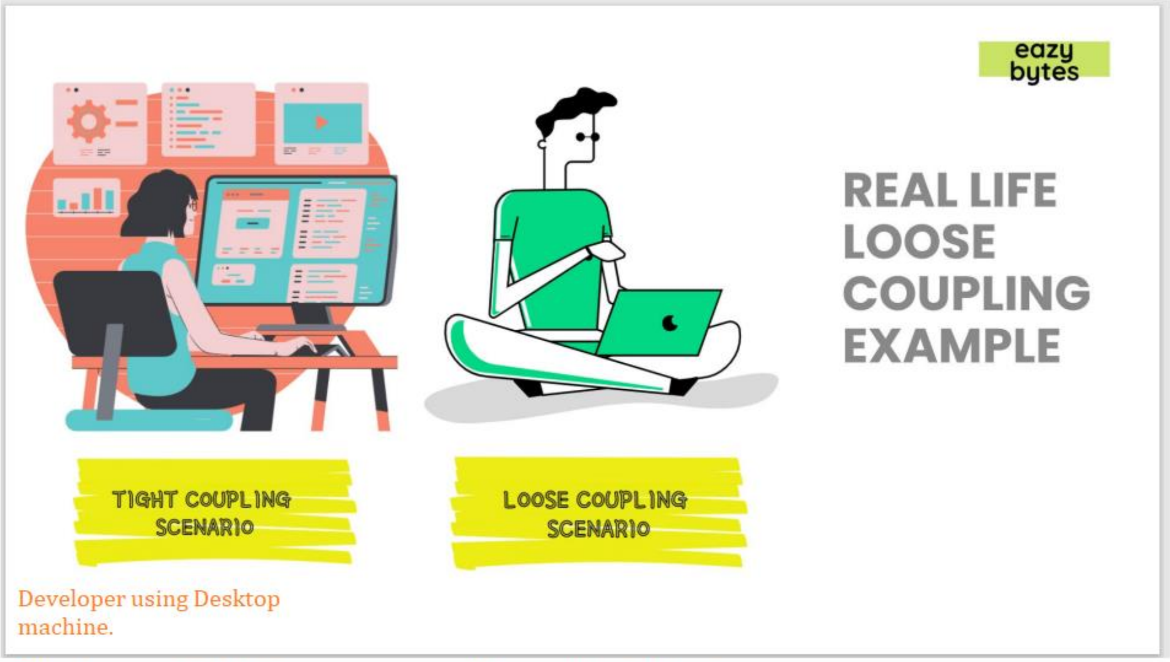
eazy bytes
LOOSE COUPLING BETWEEN THE COMPONENTS
MINIMIZES THE AMOUNT OF CODE IN YOUR APPLICATION
MAKES UNIT TESTING EASY WITH DIFFERENT MOCKS
INCREASED SYSTEM MAINTAINABILITY & MODULE REUSABILITY
ALLOWS CONCURRENT OR INDEPENDENT DEVELOPMENT
REPLACING MODULES HAS NO SIDE EFFECT ON OTHER MODULES

As the components are Loosely coupled .unit testing will be Easy.

Consider a scenario where the Speaker Vendor is not yet ready
with the speaker then as it is loosely coupled we can complete the
Unit testing with some MOCK speaker

Consider a scenario where we want to replace the vehicle with Different
set of speakers we does not have change anything in the Vehicle class
as it will be not aware of the Objects

As they are loosely coupled
even if the other
components are not ready
we can go with the
development of other
components



eazy bytes

REAL LIFE LOOSE COUPLING EXAMPLE

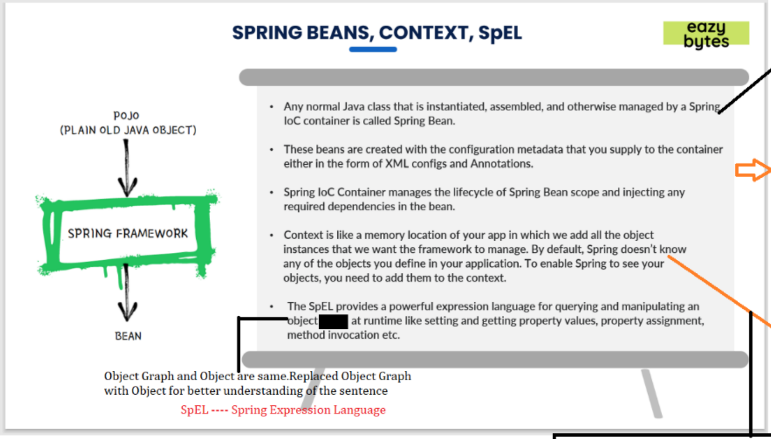
TIGHT COUPLING SCENARIO

LOOSE COUPLING SCENARIO

Developer using Desktop machine.

When they want to shift to different Work place they need to take all the required things then only they can be moved different place.

Where as if we take Laptop User there will less dependencies



SPRING BEANS, CONTEXT, SpEL

eazy bytes

POJO
(PLAIN OLD JAVA OBJECT)

SPRING FRAMEWORK

BEAN

- Any normal Java class that is instantiated, assembled, and otherwise managed by a Spring IoC container is called Spring Bean.
- These beans are created with the configuration metadata that you supply to the container either in the form of XML configs and Annotations.
- Spring IoC Container manages the lifecycle of Spring Bean scope and injecting any required dependencies in the bean.
- Context is like a memory location of your app in which we add all the object instances that we want the framework to manage. By default, Spring doesn't know any of the objects you define in your application. To enable Spring to see your objects, you need to add them to the context.
- The SpEL provides a powerful expression language for querying and manipulating an object [redacted] at runtime like setting and getting property values, property assignment, method invocation etc.

Object Graph and Object are same.Replaced Object Graph with Object for better understanding of the sentence

SpEL ---- Spring Expression Language

whenever a Java class is being maintained by the spring container, then that Java class we can call it as a bean.

As a developer, it's our responsibility to mention those details which java class has to be maintained by Spring Framework,and which Java class should not be maintained with the help of configurations that we supply to the spring IOC container, either in the form of XML configurations or annotations.

spring container will know which class to be maintained based upon the configurations that we provide and based upon the configurations that are available.Spring container instantiates them and assembles them.

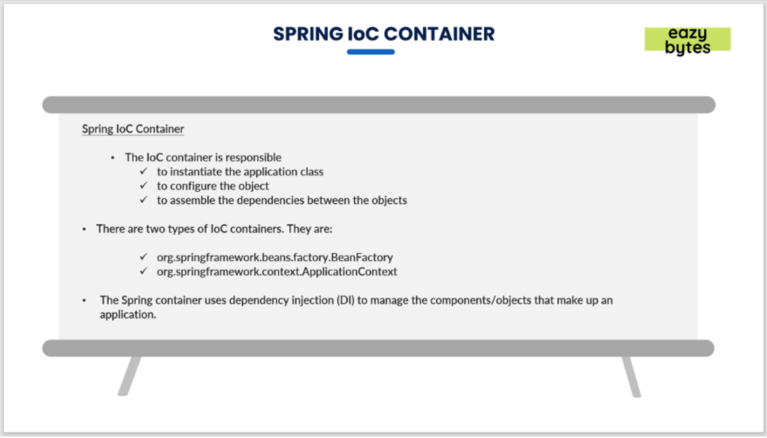
Then those Java classes will become beans inside spring framework

context is like a memory location of your [redacted] application in which all the objects that are instantiated by the framework are present and managed by it with the help of context information. By default, spring does not aware about of all your Java classes.

So based upon the configurations that we do during startup of your web application, all those objects will be created and instantiated by the spring IOC container.
And when this process happens, all those objects information will be available inside the spring context.

If an object is never available inside spring context, then there is no way spring framework to maintain that. That's why please have an understanding that context is a place where all your beans information and the dependencies which will be maintained by your spring IOC container based upon the configurations that we give to the spring framework.

Spring IOC Container:



SPRING IoC CONTAINER

Spring IoC Container

- The IoC container is responsible
  - ✓ to instantiate the application class
  - ✓ to configure the object
  - ✓ to assemble the dependencies between the objects
- There are two types of IoC containers. They are:
  - ✓ org.springframework.beans.factory.BeanFactory
  - ✓ org.springframework.context.ApplicationContext
- The Spring container uses dependency injection (DI) to manage the components/objects that make up an application.

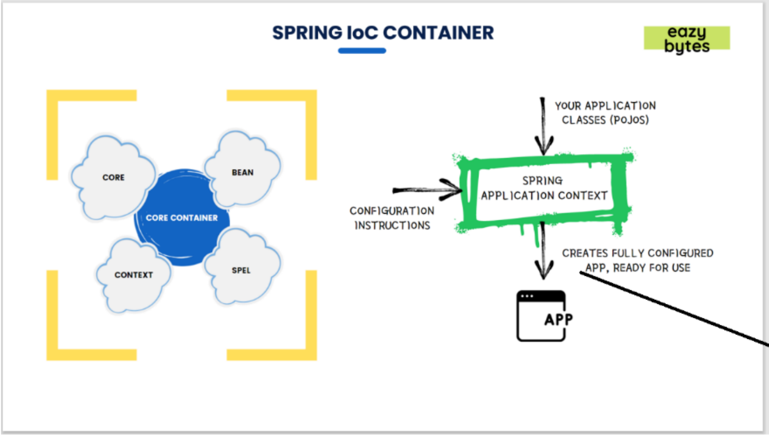
It is the IOC container that at runtime dynamically injects the dependencies between multiple objects with the dependency injection pattern that we discussed previously.

Bean Factory IOC Container :
Bean factory is a very basic IOC container where it will not provide you any advanced features, so it can only handle the bean creation, bean maintenance, auto wearing them and injecting the dependencies based upon dependency injection pattern.

Applicatiion Context IOC Container:
This is an advanced IOC container and off course application context implements bean factory.
Also,On top of that, it provides extra features to the developers.Like if you are in a scenario where after creating a bean, you want some code to be executed or just before destroying a bean if you want to execute a business logic.
So all these kind of event publishing around the spring beans can be handled with the help of application context interface and it Implementations

if you want to write a very basic spring application where you don't want to use any advanced features of spring framework, then you should go for the bean factory implementation.

Otherwise I would always recommend you to go with the application context.



SPRING IoC CONTAINER

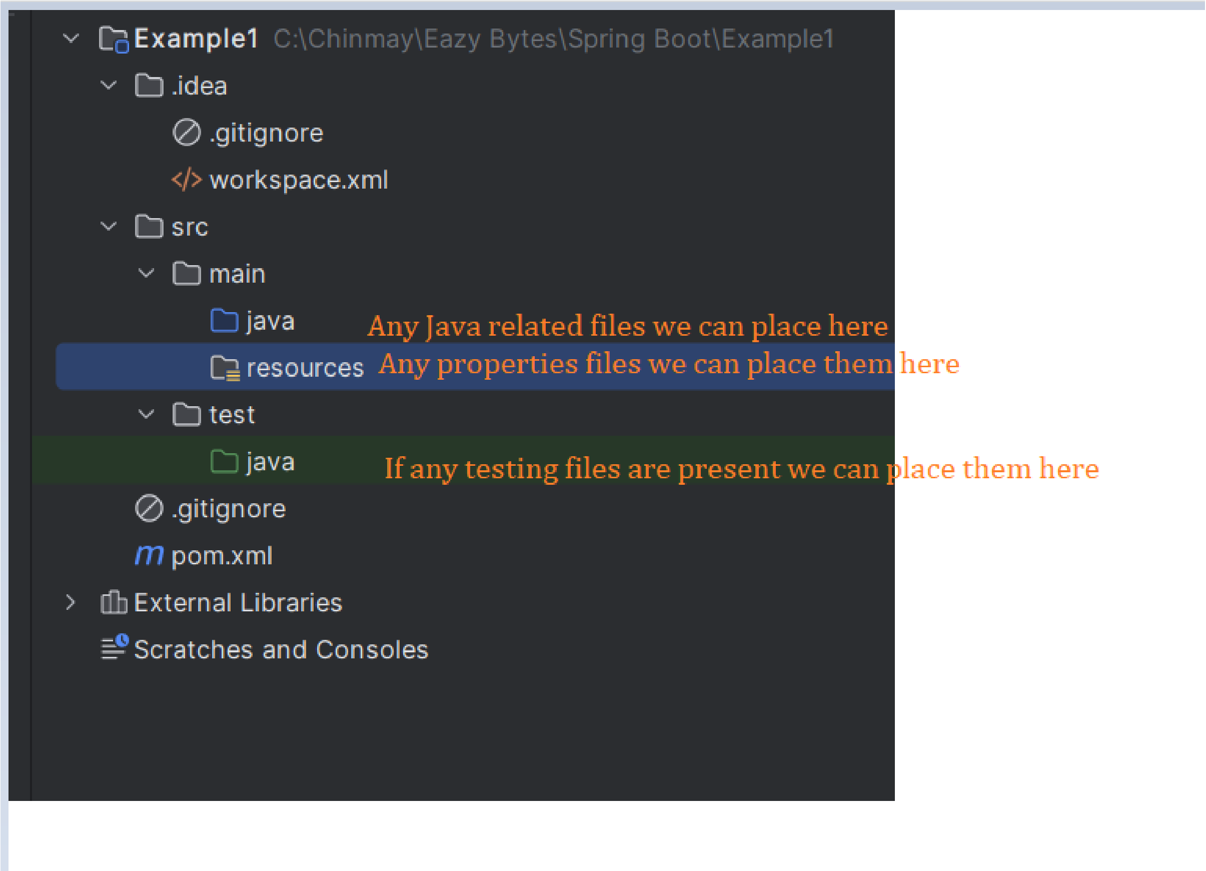
eazy bytes
CORE
BEAN
CORE CONTAINER
CONTEXT
SPEL
YOUR APPLICATION CLASSES (POJOS)
CONFIGURATION INSTRUCTIONS
SPRING APPLICATION CONTEXT
CREATES FULLY CONFIGURED APP, READY FOR USE
APP

IOC container also leverages the context, which is a virtual memory location inside spring framework where all the configurations around how to create a bean.
What are the dependencies that it has, what are the initial values that we want to instantiate? So all those information available inside a memory location called context.

So spring IOC container during the startup of the application will go ahead and look for the context information. So based upon the configurations available inside the context location, it will convict all the applicable Java objects into the beans and after converting the Java objects into the spring beans, it will also understand what are the dependencies between all these hundreds and thousands of beans are available inside a web application and during runtime, by using spring expression language, it will try to inject the dependencies based upon the configurations that we mentioned inside the context location.

for the spring application context, if you provide the list of application classes, which are your pojo classes that you have written, all your business logic and the other side, you will also provide the configuration instructions that you have, like which classes you want to convert them to beans.

What are the dependencies that a specific bean has or an object has inside your application?
What are the initial values?
So all those information you will provide using configurations instructions with the help of XML configurations or annotation configurations.
So by taking these two information, spring application context will prepare a fully configured application which is ready for use inside a production environment.



```
Example1 C:\Chinmay\Eazy Bytes\Spring Boot\Example1
  .idea
    .gitignore
    workspace.xml
  src
    main
      java
      resources
    test
      java
  .gitignore
  pom.xml
External Libraries
Scratches and Consoles
```

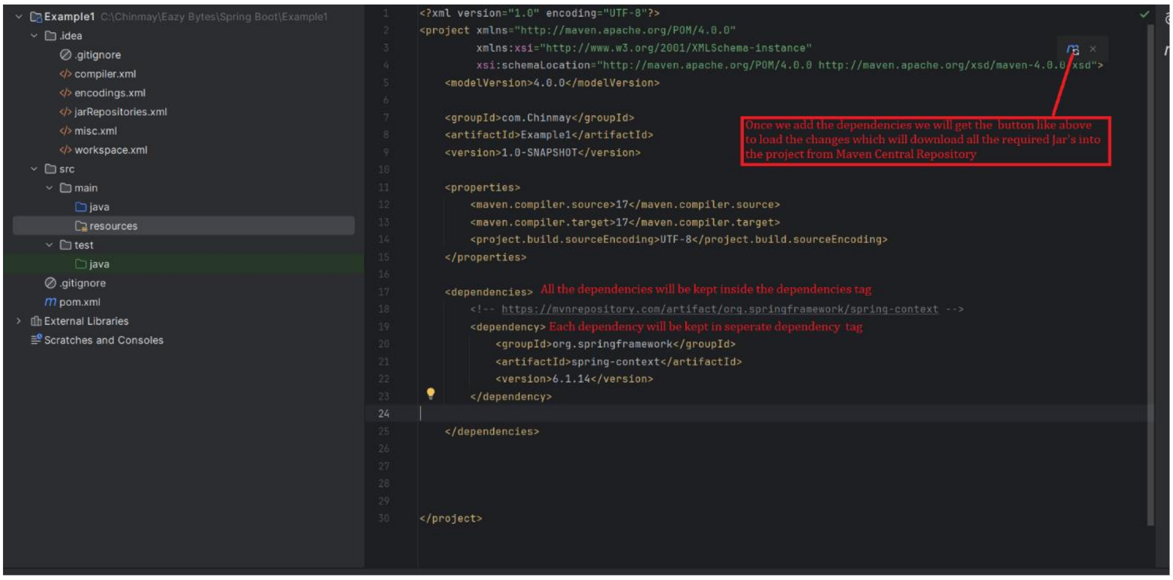
Any Java related files we can place here

Any properties files we can place them here

If any testing files are present we can place them here

In Pom.xml we will be defining the dependencies. Based on the dependencies defined corresponding Jar files will be downloaded from the Maven Central Repository.

Consider if we have shared the project to someone then we does not have to share any Jar files. As these dependencies will be already present in pom.xml once they import and load the Project the Jar files will be downloaded from the repository.



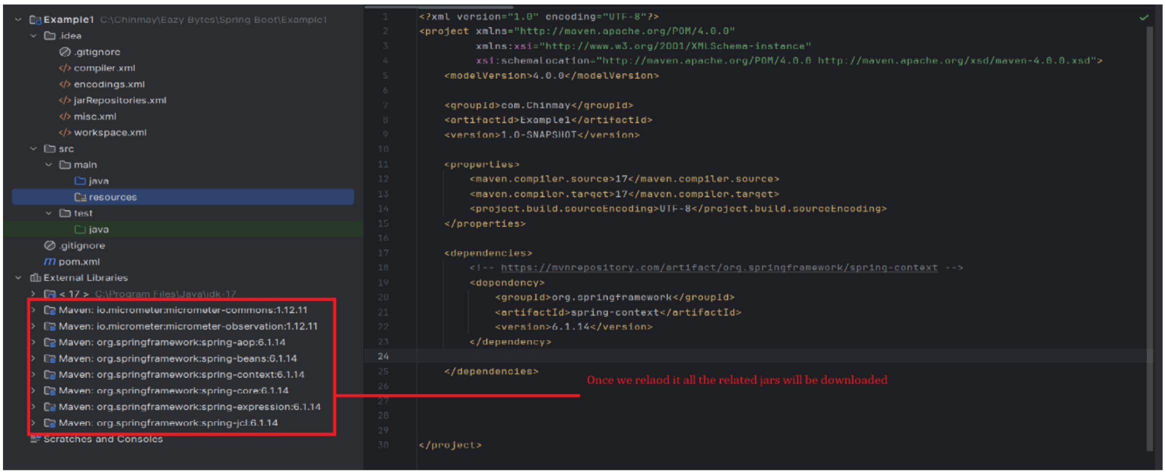
```
Example1  C:\Chinmay\Eazy Bytes\Spring Boot\Example1
  .idea
    .gitignore
    compiler.xml
    encodings.xml
    jarRepositories.xml
    misc.xml
    workspace.xml
  src
    main
      java
      resources
    test
      java
  .gitignore
  pom.xml
External Libraries
Scratches and Consoles
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.Chinmay</groupId>
    <artifactId>Example1</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>6.1.14</version>
        </dependency>

    </dependencies>




</project>
```

Once we add the dependencies we will get the button like above to load the changes which will download all the required Jar's into the project from Maven Central Repository

All the dependencies will be kept inside the dependencies tag

Each dependency will be kept in seperate dependency tag



```
Example1  C:\Chinmay\Eazy Bytes\Spring Boot\Example1
  .idea
    .gitignore
    compiler.xml
    encodings.xml
    jarRepositories.xml
    misc.xml
    workspace.xml
  src
    main
      java
      resources
    test
      java
  .gitignore
  pom.xml
External Libraries
  < 17 >  C:\Program Files\Java\jdk-17
  Maven: io.micrometer:micrometer-commons:1.12.11
  Maven: io.micrometer:micrometer-observation:1.12.11
  Maven: org.springframework:spring-aop:6.1.14
  Maven: org.springframework:spring-beans:6.1.14
  Maven: org.springframework:spring-context:6.1.14
  Maven: org.springframework:spring-core:6.1.14
  Maven: org.springframework:spring-expression:6.1.14
  Maven: org.springframework:spring-jcl:6.1.14
Scratches and Consoles
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.Chinmay</groupId>
    <artifactId>Example1</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>6.1.14</version>
        </dependency>

    </dependencies>




</project>
```

Once we reload it all the related jars will be downloaded