SPRING CORE:

- **Spring Framework Overview:**

Inversion of Control (IoC) and Spring’s Role

Inversion of Control is the principle where object creation and dependency management are handled by a container (like Spring) instead of the developer. In Spring, objects managed by the framework are called "beans." This further reduces coupling and centralizes configuration.

Overview of Spring Framework and Spring Boot

Spring is a Java-based framework introduced in 2004 by Rod Johnson (VMware), with version 1.x, now at version 6.x. Spring Boot is a tool built on top of Spring that simplifies setup and configuration, making it easier to start new projects with minimal effort and boilerplate code.

Layered Architecture in Spring Applications

Spring supports a standard three-layer architecture:

- Controller / Web layer (handles HTTP requests),
- Service / Business layer (contains business logic), and

- Persistence / DAO layer (handles database operations).
- Each layer can use different technologies, and Spring provides modules for all.

Spring Modules and Versatility

Spring offers specialized modules for different needs: Spring MVC for web, Spring Data JPA/JDBC for database, Spring Security for security, Spring Batch for batch processing, and Spring AOP for cross-cutting concerns. Spring is modular and can integrate with other frameworks like Hibernate or Struts.

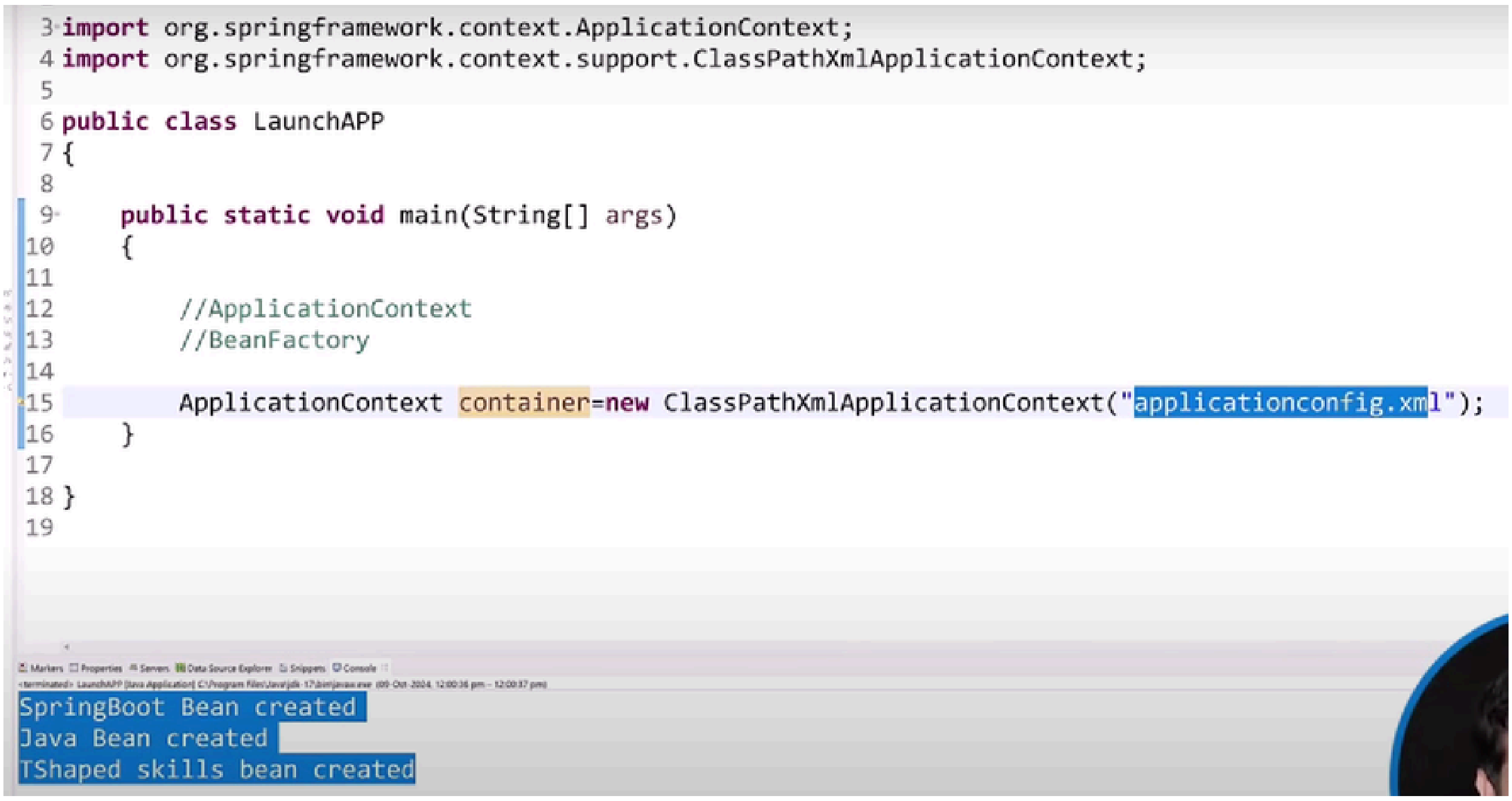
Importance of Spring Core

Spring Core, which covers dependency injection and IoC, is the foundation for all other Spring modules. Understanding it is essential before moving on to advanced features or modules.

Introduction to Maven for Dependency Management

Maven is a build tool that helps manage external libraries (like Spring) by downloading and adding them automatically to your project. Instead of manually adding jar files, you declare dependencies in a configuration file (pom.xml), and Maven handles the rest.

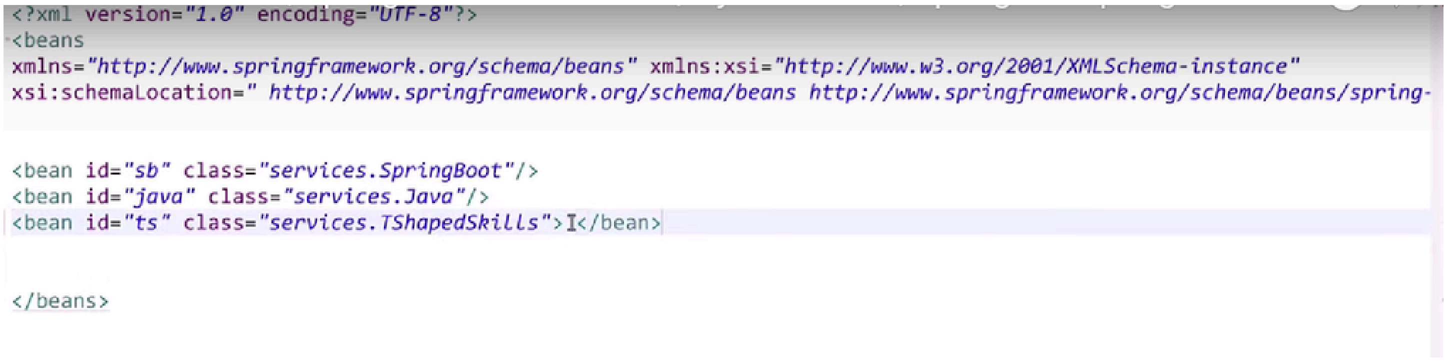
- How Spring works: (**XML based approach**)



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

public class LaunchAPP
{

    public static void main(String[] args)
    {

        //ApplicationContext
        //BeanFactory

        ApplicationContext container=new ClassPathXmlApplicationContext("applicationconfig.xml");
    }

}

```

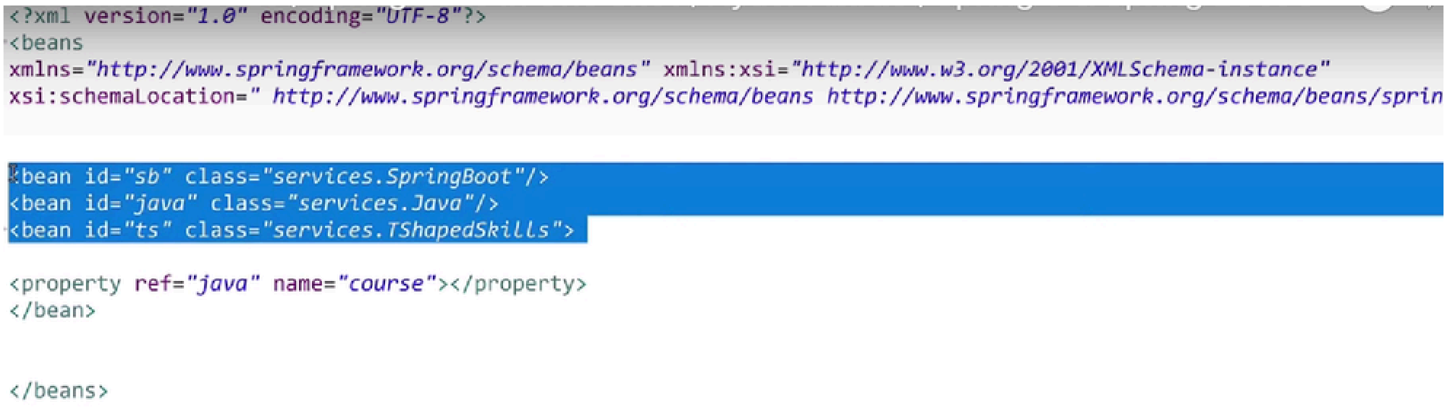
```
SpringBoot Bean created
Java Bean created
TShaped skills bean created
```

- Using the same classes (TShapedSkills, Java and SpringBoot).
- The container object of ApplicationContext is initializing the IOC(Inversion of Control) which will create object of all 3 classes.
- All 3 classes have no args constructor.
- In main method we are not using any name out of those 3 classes and yet we are able to initialise them.
- The “applicationconfig.xml” contains ->



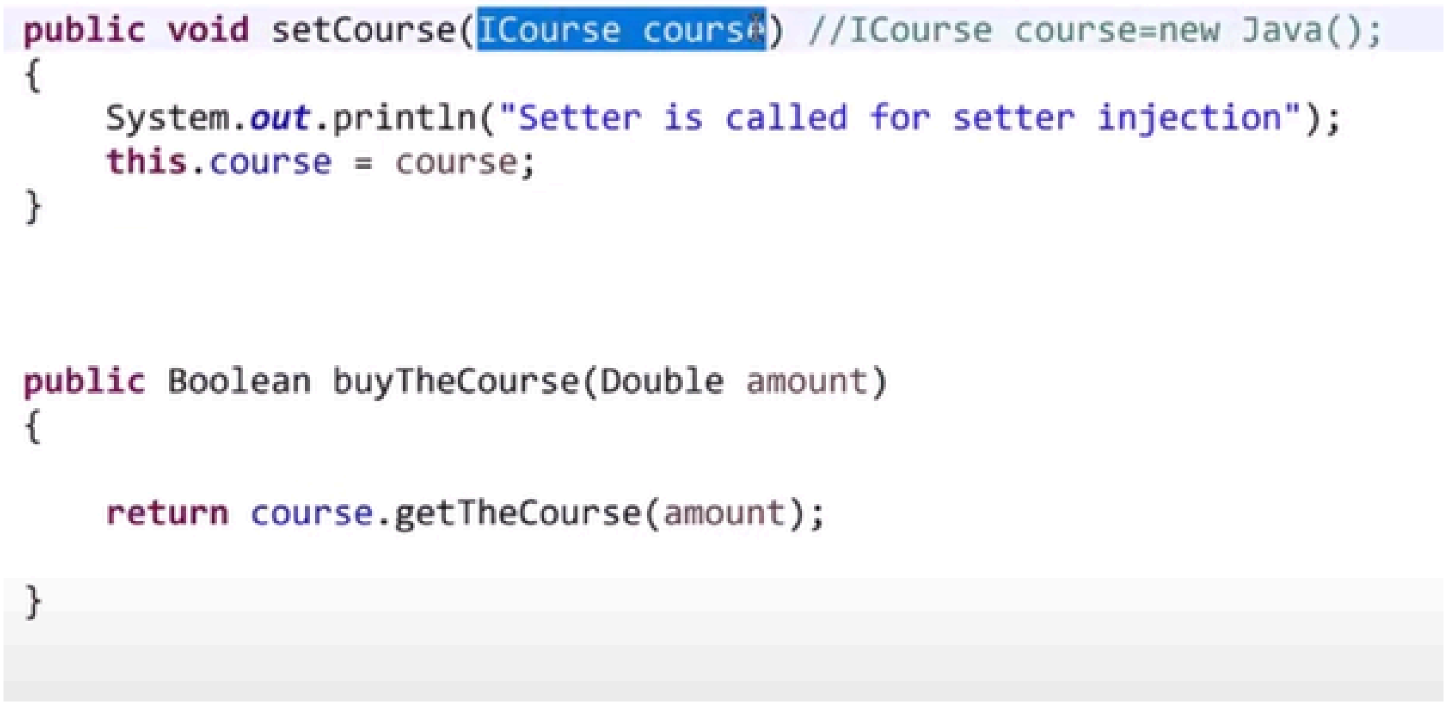
```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-

<bean id="sb" class="services.SpringBoot"/>
<bean id="java" class="services.Java"/>
<bean id="ts" class="services.TShapedSkills"></bean>


</beans>
```

- We have stated, what objects should spring make.
- Currently we are using constructor injection.
- If we want to use setter injection then, refer below image.
- ref = “java” tells what object to inject and name = “course” tells where to inject.



```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/sprin

<bean id="sb" class="services.SpringBoot"/>
<bean id="java" class="services.Java"/>
<bean id="ts" class="services.TShapedSkills">

<property ref="java" name="course"></property>
</bean>


</beans>
```

- We can see that this Java class’s object will be injected in place of course instance.



```java
public void setCourse(ICourse course) //ICourse course=new Java();
{
    System.out.println("Setter is called for setter injection");
    this.course = course;
}



public Boolean buyTheCourse(Double amount)
{

    return course.getTheCourse(amount);

}
```

We saw we can implement IOC and DI using XML.