SPRING CORE

- Create maven project
  - Packaging as jar
- Dependencies
  - Spring-context, slf4j, log4j
- Configure beans-config.xml
- Create main, service,model,dao, exceptional handling
- Create ApplicationContext supplying configuration (xml/annotaiton)
- Build and package as jar

— Why framework ?
Can use underlying platform
Faster development
Already tested and proven
Easy maintenance

— Why Spring Framwork?
Java application Development Framework
Provides POJOs (Plain Old Java Objects)
Metadata configuration + POJO = fully configured Bean/System

— History
Author- Rod Johnson. — 2002 initial release

XML Configuration - 2.0
Annotations -
Java 8 features -
Latest version - 5.3.20

Pivotal software(controls Spring initiatives) owned by Dell

— Modules in Spring
- Core container
  - Beans
  - core
  - Context
  - SpEL
- Web
  - web
  - servlet
  - socket
  - portlet
- Dataa Access/Integration

- JDBC
- ORM
- OXM
- JMS
- Transactions

- Spring AOP

- aspects, messaging, instrumentation

- Aspect Oriented Programming , based on aspects (technical features like logging, transactional checks,)

building features separately , we can plugin or plugout these

— Dependency Injection:
The framework injects the necessary dependencies when it creates the bean needed by the application
The framework constructs the objects and injected to the main program in the application.
Spring framework provides IOC container, which is carried out by DI to simplify and to configure the files and
DI is a part of IOC(Inversion of Control) container, it is one methodology to implement IoC container.

- Configuration file: can have the bean properties which constructs the respective dependent bean and the framework can configure and apply to the program to avoid repetition in the code.

- Annotations
- Java Config

— Spring Repository
https://repo.spring.io/ui/native/release/org/springframework/spring/5.3.20/

Spring Distribution
https://repo.spring.io/artifactory/release/org/springframework/spring/5.3.20/spring-5.3.20-dist.zip

— Maven - It is a build tool to add all the dependencies necessary to run an application.

- pom.xml (project object model)

- project info
- dependency management
- build management

- Maven Repository

https://mvnrepository.com/

- Maven Artifact
  - Artifact ID (collection of libraries)
  - Group ID (collection of artifacts) , usually the project name
  - the directory structure
    - src\main\java. - java programs
    - src\main\resources. - config
    - src\test\java. - for testing
    - src\test\resources - for testing
  - Maven life cycle
    - clean
    - install

— Create instance of Spring IoC Container

- using Application Context
- using Bean Factory
- diff between ApplicationContext and Bean Factory

	BeanFactory	ApplciationContext
- Bean instantiation/Wiring	Y	Y
- Integrated lifecycle management	N	Y
- automatic Bean post Processor registration	N	Y
- automatic Bean post Processor registration	N	Y
- Convenient message source access for Internalization	N	Y
- Built-in ApplicationEvent publication mechanism	N	Y

- Can we create multiple IoC containers ? Yes, we can create
- Ex:

```
ApplicationContext ctx = new ClassPathXmlApplicationContext("bean-config.xml");
//  creates spring Ioc container instance and by default creates the beans defined in the bean-config.xml file
// if lazy-init = true, it does not create bean by default, will create only when referenced
```

— Bean Instantiation

- constructor method (with default or argument , based on parameter name or datatype or index(position))

- static factory method
- instance factory method

— Dependency Injection types
- Constructor injection
  It includes dependencies in constructor as arguments, mandatorily, you cannot change the dependencies once passed to the constructor (implements immutable objects)
- Setter injection
  It includes the dependencies with setter method as arguments , but not mandatory, can change the dependencies when needed.
  (creates mutable objects)

— Bean
- -Scopes
  - singleton(default) - single object created for IoC Container
  - prototype - each reference creates new instance for the container
  - request -
  - session
  - application
- Lazy initialization
  - By default, Spring creates eagerly all the bean definitions
  - Mark lazy-init = true attribute in the bean definition, it will be initialized only when invoked.

  Optimises boot up speed of the application having more beans

- Bean life cycle methods (Init, destroy)
  - init or InitializingBean interface
  - cleanup or DisposableBean
  - by default, the init(), destroy() methods are invoked by the container
  - if we have diff names for these methods give them like this <bean id “” init-method = “initialize” destroy-method = “cleanup” />.
  - BeanPostProcessor interfcae
    - To allow custom modification of new bean instances
    - To add additional functinality to all the beans explicitly
      - postProcessBeforeInitialization() - called before init() method
      - postProcessAfterInitialization() - called after init() method
- Depends On
  - If bean1 depends on bean2, bean1 will be constructed, only after bean2 is constructed
  - we specify in bean properties
- Abstract bean : Bean Inheritance

- Bean definition is not done for abstract bean
- it can be inherited by other same bean definitions
- If 1 variable is common cross all the bean instances, extract that and make it as abstract bean and use it across all the bean instances.
- It will not change the class level bean properties(variables), the inheritance is done only wile defining the bean (can do in xml)
- The matching between the parent and the child can be done by name or by type
- person

— Autowiring
Autowiring feature of spring framework **enables you to inject the object dependency implicitly**. It internally uses setter or constructor injection. Autowiring can't be used to inject primitive and String values. It works with reference only.

Spring can Autowire relationships between objects
No need to inject the dependency explicitly using

- Autowiring modes.
- by default no auto wiring, it does setter injection
- if given autowire, Autowired by type by default
- by name
- by type
- by constructor

- Primary Attribute: If more than 1 bean found matching either by name or type, we have primary attribute set to true
- Auto-wireIf bean attribute auto wire-candidate = false, to disable auto wiring for that bean

- pros of autowire: no need to explicitly provide the dependency details
- cons of autowire: explicit dependency injection by constructor or setter overrides auto wiring

— Configuration done by 3 ways
- XML Based
- cons:
- more XML configuration
- annotation based
- cons:

- can’t create multiple instances of bean
- annotations be scattered across all the classes
- requires rebuild and deployment if any changes

- Java based
  - create configuration classes to

— employee-mgmt-with-spring-core. (Java based configuration)

Used @Bean, @Configuration in Config class