# Spring Boot Dependency Injection

# Agenda

1. Dependency Injection
2. Singleton Bean Scope
3. Prototype Bean Scope
4. Autowiring

# Objectives

At the end of this module, you will be able to:

- Understand, what is dependency injection in spring boot.
- Understand, what is auto wiring in spring boot.

# Dependency Injection

# What is Dependency Injection?

- Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container "injects" objects into other objects or "dependencies".

- Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

# Dependency Injection Scenarios.

**Class Employee**

Department dep;

Project proj;

**Class Department**

**Class Project**

**Class Laptop**

HardDisk hd;

Battery battery;

**Class HardDisk**

**Class Battery**

# Spring Boot Dependency Injection.

- Spring Container:

  When we are running a Spring Boot application, by default it will create the Spring Container inside a JVM.

  SpringApplication.run() will create a Spring Container. And it will return an object of "ConfigurableApplicationContext".

- @Component:

  This annotation will tell the Spring Boot to create an object of particular bean class inside the Spring Container.

- getBean():

  By using "ConfigurableApplicationContext" object, we can call a method getBean() to get a particular bean object from the Spring Container.

# Dependency Injection - Example

# Spring Boot Dependency Injection.

```
package com.wipro.pack;
import org.springframework.stereotype.Component;
@Component
public class Employee {
            private int eid;
            private String ename;
            public int getEid() {
                        return eid;
            }
            public void setEid(int eid) {
                        this.eid = eid;
            }
            public String getEname() {
                        return ename;
            }
            public void setEname(String ename) {
                        this.ename = ename;
            }
            public void display() {
                        System.out.println("You got Employee object");
            }
}
```

# Spring Boot Dependency Injection.

```
package com.wipro.pack;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
@SpringBootApplication
public class SbDependencyInjectionApplication {

            public static void main(String[] args) {
           ConfigurableApplicationContext context = SpringApplication.run(SbDependencyInjectionApplication.class, args);
           Employee emp1 = context.getBean(Employee.class);
           emp1.display();
            }
}
```

Console | Progress | Problems

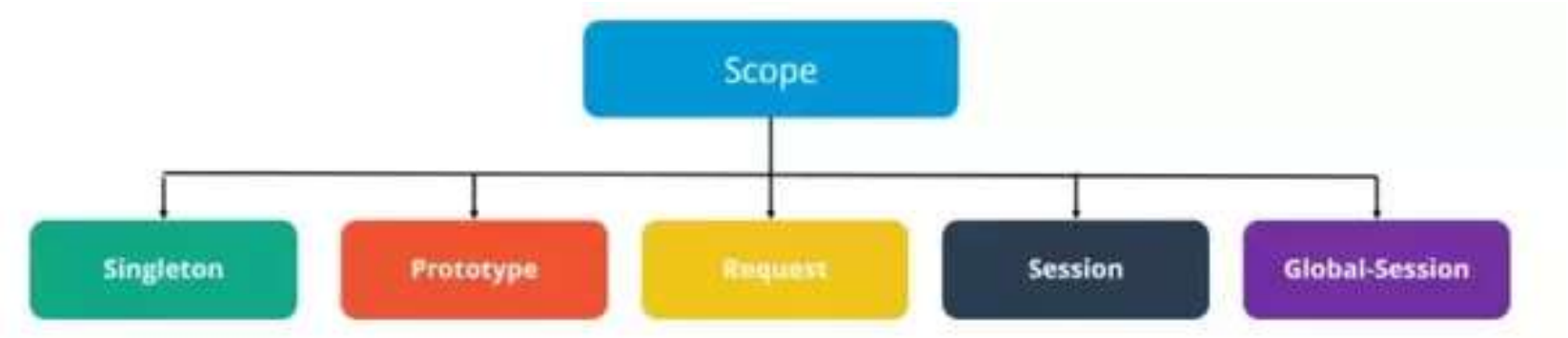
SB-DependencyInjection - SbDependencyInjectionApplication [Spring Boot App]

```
2020-04-25 19:28:48.390  INFO 24500 --- [          main] trationDelegate$BeanPostProcessorChecker : Bean
2020-04-25 19:28:48.484  INFO 24500 --- [          main] .w.s.a.s.AnnotationActionEndpointMapping : Supp
2020-04-25 19:28:48.807  INFO 24500 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2020-04-25 19:28:48.817  INFO 24500 --- [          main] o.apache.catalina.core.StandardService   : Star
2020-04-25 19:28:48.817  INFO 24500 --- [          main] org.apache.catalina.core.StandardEngine  : Star
2020-04-25 19:28:49.212  INFO 24500 --- [          main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Init
2020-04-25 19:28:49.213  INFO 24500 --- [          main] o.s.web.context.ContextLoader            : Root
2020-04-25 19:28:49.479  INFO 24500 --- [          main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Init
2020-04-25 19:28:49.773  INFO 24500 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2020-04-25 19:28:49.780  INFO 24500 --- [          main] c.w.p.SbDependencyInjectionApplication   : Star
You got Employee object
```

# Spring Boot Bean Scope

# Spring Boot Bean Scope

The Scope of a bean is a metadata where we can specify which Instance we would like to get from the container. In Spring we have the opportunity to choose among five different bean scopes.



# Spring Boot Bean Scope

- **Singleton scope**
  - This is the default scope. It means that the IoC container will only create exactly one instance of the object defined by that bean definition. The container stores this particular instance to a cache. Therefore all request which points to that bean will get this single instance.
- **Prototype scope**
  - If you define a bean as a prototype the IoC container will serve a new instance from that bean every time you call for it.
- **Request scope**
  - The web container creates a new instance for every independent HTTP request. Hence, they destroy every time when the call ends.

# Spring Boot Bean Scope

- **Session scope**
  - The container returns a new instance for every session. Hence if we call our controller in the same Session the result will be the same.
- **Global-Session scope**
  - The global session scoped bean instance is shared across your web application. Hence every call receives the same bean instance. Its similar to the singleton in normal core applications.

# Singleton Bean Scope

# Singleton Bean Scope.

- Singleton scope in the spring framework is the default bean scope in the spring container.
- It tells the container to exactly create a single instance of the object.
- This single instance is stored in the cache and all the subsequent requests for that named bean return the cached instance.
- Even if there is no request raised for a bean object also it will create one instance for the particular bean class.

# Singleton Bean Scope - Example

# Singleton Bean Scope.

```
package com.wipro.pack;
import org.springframework.stereotype.Component;
@Component
public class Employee {
            private int eid;
            private String ename;
            public Employee() {
                        System.out.println("Inside Employee Constructor");
            }
            public int getEid() {
                        return eid;
            }
            public void setEid(int eid) {
                        this.eid = eid;
            }
            public String getEname() {
                        return ename;
            }
            public void setEname(String ename) {
                        this.ename = ename;
            }
            public void display() {
                        System.out.println("You got Employee object");
            }
}
```

# Singleton Bean Scope.

```
package com.wipro.pack;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
@SpringBootApplication
public class SbDependencyInjectionApplication {

            public static void main(String[] args) {
          ConfigurableApplicationContext context = SpringApplication.run(SbDependencyInjectionApplication.class, args);
          Employee emp1 = context.getBean(Employee.class);
          emp1.display();
          Employee emp2 = context.getBean(Employee.class);
          emp2.display();
            }
}
```

```
Console   Progress   Problems
SB-DependencyInjection - SbDependencyInjectionApplication [Spring Boot App] C:\Program Files\Java\jdk1.8.0_40\bin\javaw.exe  (Apr 25, 2
2020-04-25 19:45:15.976  INFO 15132 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2020-04-25 19:45:15.985  INFO 15132 --- [           main] o.apache.catalina.core.StandardService   : Star
2020-04-25 19:45:15.985  INFO 15132 --- [           main] org.apache.catalina.core.StandardEngine  : Star
2020-04-25 19:45:16.293  INFO 15132 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Init
2020-04-25 19:45:16.293  INFO 15132 --- [           main] o.s.web.context.ContextLoader            : Root
Inside Employee Constructor
2020-04-25 19:45:16.544  INFO 15132 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Init
2020-04-25 19:45:16.805  INFO 15132 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2020-04-25 19:45:16.810  INFO 15132 --- [           main] c.w.p.SbDependencyInjectionApplication   : Star
You got Employee object
You got Employee object
```

# Prototype Bean Scope

# Prototype Bean Scope

- If the scope is set to prototype, the Spring Container creates a new bean instance of the object every time a request for that specific bean is made.

- The spring container will create an instance for a particular bean class if the request is raised. Other wise no instance will be created for the particular bean class.

# Prototype Bean Scope - Example

# Prototype Bean Scope.

```
package com.wipro.pack;
import org.springframework.stereotype.Component;
@Component
@Scope("prototype")
public class Employee {
            private int eid;
            private String ename;
            public Employee() {
                        System.out.println("Inside Employee Constructor");
            }
            public int getEid() {
                        return eid;
            }
            public void setEid(int eid) {
                        this.eid = eid;
            }
            public String getEname() {
                        return ename;
            }
            public void setEname(String ename) {
                        this.ename = ename;
            }
            public void display() {
                        System.out.println("You got Employee object");
            }
}
```

# Prototype Bean Scope.

```
package com.wipro.pack;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
@SpringBootApplication
public class SbDependencyInjectionApplication {

            public static void main(String[] args) {
          ConfigurableApplicationContext context = SpringApplication.run(SbDependencyInjectionApplication.class, args);
          Employee emp1 = context.getBean(Employee.class);
          emp1.display();
          Employee emp2 = context.getBean(Employee.class);
          emp2.display();
            }
}
```

Console  Progress  Problems

SB-DependencyInjection - SbDependencyInjectionApplication [Spring Boot App]

```
2020-04-25 19:58:07.241  INFO 10996 --- [           main] o.apache.catalina.core.StandardService   : Sta
2020-04-25 19:58:07.241  INFO 10996 --- [           main] org.apache.catalina.core.StandardEngine  : Sta
2020-04-25 19:58:07.568  INFO 10996 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Ini
2020-04-25 19:58:07.569  INFO 10996 --- [           main] o.s.web.context.ContextLoader            : Roo
2020-04-25 19:58:07.936  INFO 10996 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Ini
2020-04-25 19:58:08.191  INFO 10996 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tom
2020-04-25 19:58:08.195  INFO 10996 --- [           main] c.w.p.SbDependencyInjectionApplication   : Sta
Inside Employee Constructor
You got Employee object
Inside Employee Constructor
You got Employee object
```

# Autowiring

# What is Autowiring?

- Autowiring happens by placing an instance of one bean into an instance of another bean.
- Both classes should be beans.
- i.e. They should be defined to live in the application context or spring container.

# Autowiring - Example

# Autowiring – Example.

```java
package com.wipro.pack;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Employee {
            private int eid;
            private String ename;
            @Autowired
            private Department dep;
        //Generate Getters & Setters of eid & ename
            public Department getDep() {
                        return dep;
            }
            public void setDep(Department dep) {
                        this.dep = dep;
            }
            public void display() {
            System.out.println("Got Employee Object");
            dep.display();
            }
}
```

```java
package com.wipro.pack;
import org.springframework.stereotype.Component;
@Component
public class Department {
            private int did;
            private String dname;
            public int getDid() {
                        return did;
            }
            public void setDid(int did) {
                        this.did = did;
            }
            public String getDname() {
                        return dname;
            }
            public void setDname(String dname) {
                        this.dname = dname;
            }
            public void display() {
            System.out.println("Got Department Object");
            }
}
```

# Autowiring – Example.

```
package com.wipro.pack;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
@SpringBootApplication
public class SbAutowiringApplication {
            public static void main(String[] args) {
            ConfigurableApplicationContext context = SpringApplication.run(SbAutowiringApplication.class, args);

            Employee emp = context.getBean(Employee.class);
            emp.display();
            }
}
```

Console  Progress  Problems

SB-Autowiring - SbAutowiringApplication [Spring Boot App]

```
2020-04-25 22:10:28.860  INFO 8300 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initi
2020-04-25 22:10:28.861  INFO 8300 --- [           main] o.s.web.context.ContextLoader            : Root
2020-04-25 22:10:29.075  INFO 8300 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initi
2020-04-25 22:10:29.334  INFO 8300 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomca
2020-04-25 22:10:29.338  INFO 8300 --- [           main] com.wipro.pack.SbAutowiringApplication   : Start
Got Employee Object
Got Department Object
```

# Dependency Injection Conclusion.

**@Component**
**Class Employee**

```
@Autowired
Department dep;
@Autowired
Project proj;
```

**@Component**
**Class Department**

**@Component**
**Class Project**

**@Component**
**Class Laptop**

```
@Autowired
HardDisk hd;
@Autowired
Battery battery;
```

**@Component**
**Class HardDisk**

**@Component**
**Class Battery**

# Summary

In this session, you have learned about:

- What is dependency injection in spring boot.
- What is auto wiring in spring boot.

# Thank you