

# what we will learn in this video?

## AnnotationConfigApplicationContext

@Component

@ComponentScan

@Bean

@Autowired

@Value

@Primary

@Qualifier

@Configuration

@Required



```
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
```

imagine this your container : - **Application Context**



Beans.xml

```
<context:component-scan  
base-package="package1" />
```

package1

```
@component("c1")  
class class1  
{  
    //yes  
}
```

```
class class2  
{  
    //no  
}
```

```
@component("c3")  
class class3  
{  
    //yes  
}
```

do you have @component ?

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

```
public void selectAllRows() throws ClassNotFoundException, SQLException {
    System.out.println("Retrieving all students data..");
    // load driver
    Class.forName(driver);

    // get a connection
    Connection con = DriverManager.getConnection(url, userName, password);

    // execute query
    Statement stmt = con.createStatement();

    ResultSet rs = stmt.executeQuery("SELECT * FROM ESNew.HostelStudentInfo");

    while (rs.next()) {
        int studentId = rs.getInt(1);
        String studentName = rs.getString(2);
        double hostelFees = rs.getDouble(3);
        String foodType = rs.getString(4);

        System.out.println(studentId + " " + studentName + " " + hostelFees + " " + foodType);
    }

    con.close();
}
```

## selectAllRows()

21:08 / 1:20:52

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

```
public void deleteStudentRecord(int studentId) throws ClassNotFoundException, SQLException {
    // load driver
    Class.forName(driver);

    // get a connection
    Connection con = DriverManager.getConnection(url, userName, password);

    // execute query
    Statement stmt = con.createStatement();

    stmt.executeUpdate("delete from ESNew.HostelStudentInfo where Student_id = " + studentId);

    System.out.println("Record deleted with the id " + studentId);

    //closing the connection
    con.close();
}
```

## deleteStudentRecord()

Ad in 1

21:18 / 1:20:52

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

```
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);
}
```

**Hey spring , Once you crate  
StudentDAO Object Please call  
**createStudentDBConnection()** by  
yourself. Don't wait for me to call it.**

36:40 / 1:20:52

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

```
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);
}
```

**We can give our init method name as  
anything. We may say it **init** or we may  
say it **createStudentDBConnection()** or  
**xyz()****

40:50 / 1:20:52

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

```

    <terminated> Test (3) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_91.jdk/Contents/Home/bin,
    setting driver..
    setting url..
    setting username..
    setting password..
    inside the custom init method
    creating connection..
    Sun Apr 28 14:38:41 IST 2019 WARN: Establishing SSL connection without server's id
    Retrieving all students data..
    1 Abhilash 200.2 non-veg
    2 Priya 1800.0 non-veg
    3 Ramya 900.3 Veg
    4 Maggi 700.7 non-veg
    6 Sunil 19000.5 veg
  
```

46:36 / 1:20:52

REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

You can add custom code/logic during bean initialization

It can be used for setting up resources like db/socket/file etc

```

    65 public void createStudentDBConnection() throws ClassNotFoundException, SQLException {
    66
    67     System.out.println("creating connection..");
    68     // load driver
    69     Class.forName(driver);
    70
    71     // get a connection
    72     con = DriverManager.getConnection(url, username, password);
    73
    74     // execute query
    75     Statement stmt = con.createStatement();
  
```

why init()??

48:55 / 1:20:52




REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

destroy method will be called before the bean is removed from the container.

```
90
91
92
93
94
95
96
97
98 public void deleteStudentRecord(int studentId) throws ClassNotFoundException, SQLException {
99
100     // execute query
101     Statement stmt = con.createStatement();
102
103     stmt.executeUpdate("delete from ESNew.HostelStudentInfo where Student_id = " + studentId);
104
105     System.out.println("Record deleted with the id " + studentId);
106
107
108
109
110
111
112 public void closeConnection() throws SQLException
113 {
114     //closing the connection
115     con.close();
116 }
```

49:59 / 1:20:52




REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...

# @PreDestory

over the method which has  
to be called before the  
container is destroyed.

51:00 / 1:20:52



**Before your container object will destroy, spring will call your custom destroy method.**



```
@PreDestroy  
public void closeConnection() throws SQLException  
{  
    //clean up job  
    //closing the connection  
    con.close();  
}
```

**Before spring remove studentDAO bean(object) from the container,It will call this method**



## Standalone app

// creating container object manually

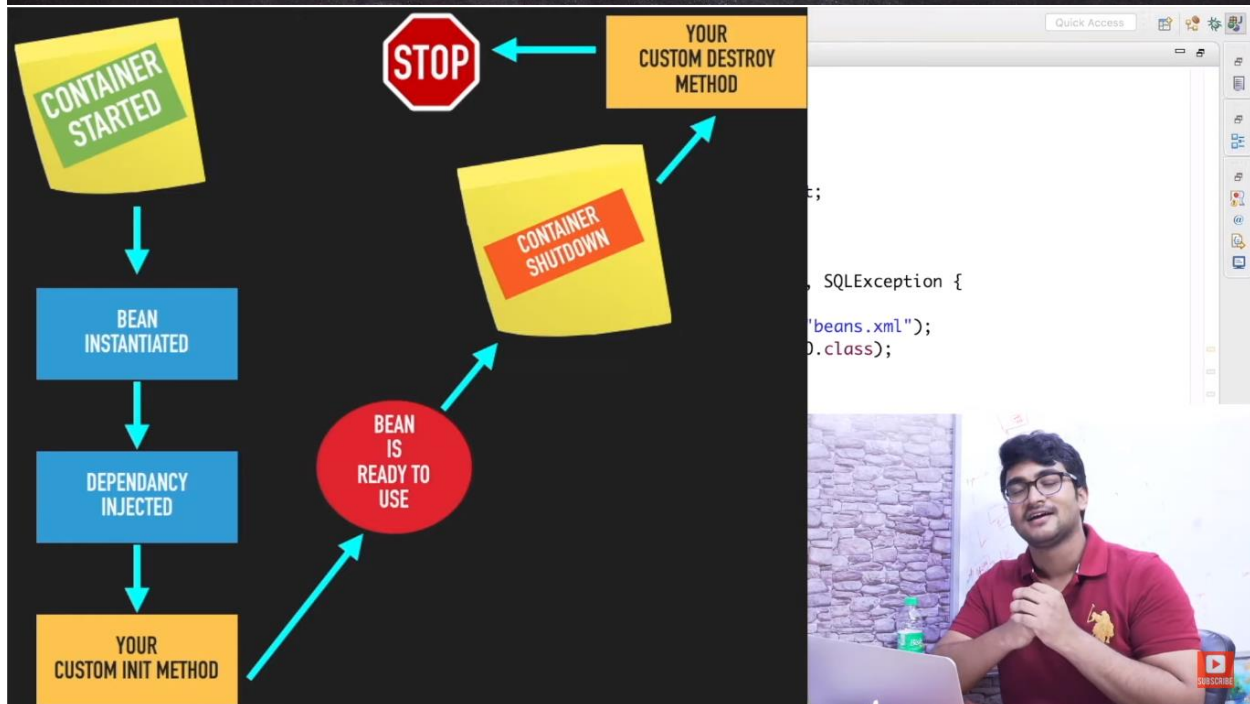
```
ApplicationContext context = new ClasspathXmlApplicationContext();
```

// destroying container object manually

```
context.close();
```




## Web App

you don't need to create and destroy the container object. This will be automatically done. We will explore more while studying spring mvc.



registerShutdownHook() will execute once the main thread ends.  
so once all your codes gets executed, It will be called and close your  
container. So It won't give us any exception irrespective of the line no  
we are calling it.



REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destro...   

Another example of using registerShutdownHook() by using  
`AbstractApplicationContext`



1:07:22 / 1:20:52





REAL-TIME: Spring Bean life cycle using @PostConstruct |@PreDestroy |init - method| destory...

StudentDAO.java

```
1 package com.seleniumexpress.beanlifecycle.annotation;
```

ApplicationContext

INTERFACE

AbstractApplicationContext

ABSTRACT CLASS

ClassPathXmlApplicationContext

CLASS

AbstractApplicationContext has a registerShutdownHook()

1:08:12 / 1:20:52

SO WE HAD TO DEFINE INIT AND DESTROY FOR 3 TIMES

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6                           http://www.springframework.org/schema/beans
7                           http://www.springframework.org/schema/context
8                           http://www.springframework.org/schema/context.xsd">
9
10  <bean id = "hello" class="com.seleniumexpress.college.Hello" init-method="init" destroy-method="destory">
11
12  <bean id = "hi" class="com.seleniumexpress.college.Hi" init-method="init" destroy-method="destory">
13
14  <bean id = "bye" class="com.seleniumexpress.college.Bye" init-method="init" destroy-method="destory">
15
16
17
18 </beans>
```

1:08:12 / 1:20:52

