Add instructor notes here.



Explain the lesson coverage

Lesson Objectives



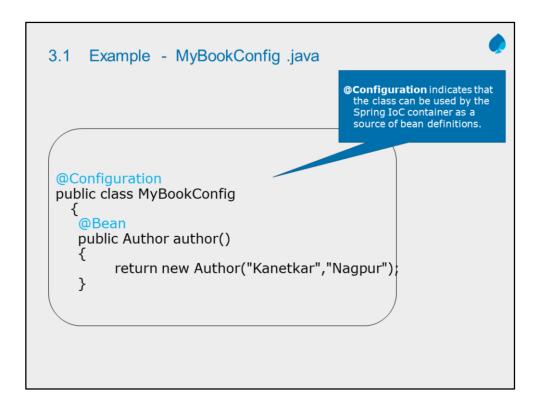
- Develop Spring Application Using Java Base Configuration
- Java Configuration Class
 - @Configuration
 - @Bean
- Implementing Bean Lifecycle Callbacks and Bean Scope
- Registering Configuration Using AnnotationConfigApplicationContext

While there are several other Java expression languages available, OGNL, MVEL, and JBoss EL, to name a few, the Spring Expression Language was created to provide the Spring community with a single well supported expression language that can be used across all the products in the Spring portfolio.

3.1 Develop Spring Application Using Java Base Configuration



- Spring 3 onwards a spring application can be configured with almost no XML using pure java.
- Java base configuration allows moving bean definition and spring configuration out of XML file into a java classes.



Annotating a class with the **@Configuration** indicates that the class can be used by the Spring IoC container as a source of bean definitions.

```
@Bean annotation tells Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context

@Bean(initMethod="setUp",destroyMethod="cleanUp")
//@Scope("prototype")
public Book book()
{

Book book=new Book();
book.setYear("1995");
book.setIsbn("Kj77756");
book.setAuthor(author());
return book;
}

}
```

3.2 Java Configuration Class



- @Configuration The java base equivalent to <beans> in xml is a Java class annotated with @Configurations
- @ Bean -This annotation is used to define the beans.

3.3:Implementing Bean Lifecycle Callbacks And Bean Scope

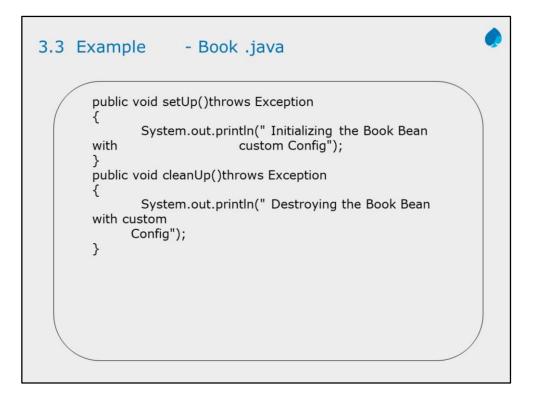


- @PostConstruct This Annotation is used on a method that needs to be executed after dependency injection is done to perform any initialization.
- @PreDestroy This Annotation is used on methods as a callback notification to signal that the instance is in the process of being removed by the container.
- @Scope This annotation is used in java base configuration to define the scope of the bean

Note:-The @PostConstruct and @PreDestroy annotation are not belong to Spring, it's located in the J2ee library - commonannotations.jar.

```
package com.cg.bean;
import javax.annotation.*;
public class Author{
    private String authorName;
    private String getAuthorName() {return authorName;}
    public void setAuthorName(String authorName) {
        this.authorName = authorName;}
        public String getAddress() {return address;}
        public void setAddress(String address) {
            this.address = address;}
}
```

3.3 Example - Author .java public Author(String authorName, String address) super(); this.authorName = authorName; this.address = address;@Override public String toString() { return "Author [authorName=" + authorName + ", address=" + address+ "]";} @PostConstruct public void customAuthorInit() System.out.println("Method customAuthorInit() invoked..."); @PreDestroy public void customAuthorDestroy() { System.out.println("Method customAuthorDestroy() invoked..."); }}



```
3.3 Example - Book .java
       public Author getAuthor() {      return author;    }
               public void setAuthor(Author author) {this.author =
       author;}
               public String getIsbn() {
                                             return isbn;
               public void setIsbn(String isbn) {this.isbn = isbn;
               public String getYear() {return year; }
               public void setYear(String year) {this.year = year;
               @Override
               public String toString()
                      return "Book [author=" + author + ", isbn=" +
                      year=" + year
                                            + "]";
       isbn + "
               }}
```

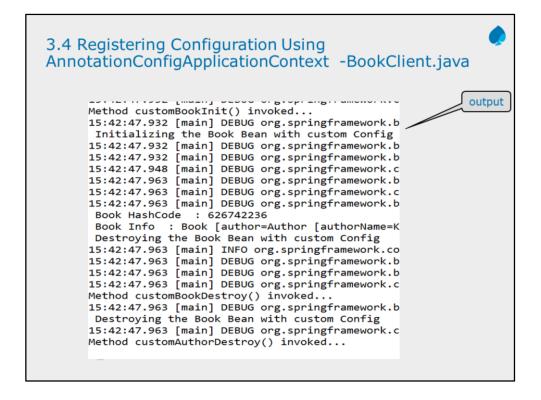
3.4 Registering Configuration Using AnnotationConfigApplicationContext -BookClient.java package com.cg.bean; import org.springframework.context.ApplicationContext; org.spring framework.context.annotation. Annotation Config ApplicationContext; public class BookClient { public static void main(String[] args) ApplicationContext ctx= new AnnotationConfigApplicationContext(MyBookConfig.class); Book book1=(Book)ctx.getBean("book"); System.out.println(" Book HashCode : "+book1.hashCode()); System.out.println(" Book Info : "+book1); try { book1.cleanUp(); catch (Exception e) { e.printStackTrace(); }}

Bean in spring can be created in java file instead of XML,too. They are created with the help of @Configuration. And that application configuration is loaded by AnnotationConfigApplicationContext in spring container.

AnnotationConfigApplicationContext(MyBookConfig.class);
Create a new AnnotationConfigApplicationContext, deriving bean definitions from the given annotated classes and automatically refreshing the context.

ApplicationContext ctx= new

- AnnotationConfigApplicationContext has the register method as well that accepts the bean configuration class.
- AnnotationConfigApplicationContext provides the method getBean to get the bean object.



```
3.4 Registering Configuration Using Boot -
BookClient.java
   //@Configuration
   //@ComponentScan
    //@EnableAutoConfiguration
    @SpringBootApplication
   public class BookClient
      public static void main(String[] args)
       ApplicationContext ctx= SpringApplication.run(BookClient.class,
       System.out.println("Welcome To Spring Boot Applications");
        Book book1=(Book)ctx.getBean("book");
        System.out.println(" Book HashCode : "+book1.hashCode());
        System.out.println(" Book Info : "+book1);
       try
            book1.cleanUp(); }
         catch (Exception e) {
            e.printStackTrace(); }}
```

- @EnableAutoConfiguration annotation auto-configures the beans that are
 present in the classpath. This simplifies the developers work by guessing
 the required beans from the classpath and configure it to run the application.
 This annotation is part of the spring boot project.
- With the spring boot 1.2.0 release, the need for this annotation has been reduced because there is an alternative annotation
 @SpringBootApplication which combines the three annotations
 @Configuration, @EnableAutoConfiguration and @ComponentScan.

Additional notes for instructor



Ans-1 : Option 1 Ans-2 : Option 3

Review Questions



- Option1:@Configuration
- Option2:@Bean
- Option3:@Component

Question 2: Once your configuration classes are defined, you can load and provide them to Spring container using ______.



- Option 1: ConfigApplicationContext
- Option 2: AnnotationApplicationContext
- Option3: AnnotationConfigApplicationContext