

Book
String name
String subject
int price
Author author

Author
String authorname
String address

Book HAS Author
Book using Author
Book depending on Author

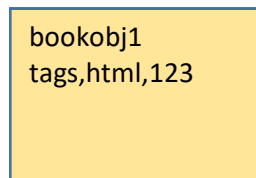
Book is Dependent

Author is Dependency

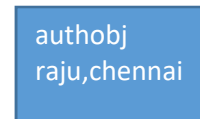
Dependency Injection

injecting dependency object into dependent object
injecting author object into book object

bo1



ao1



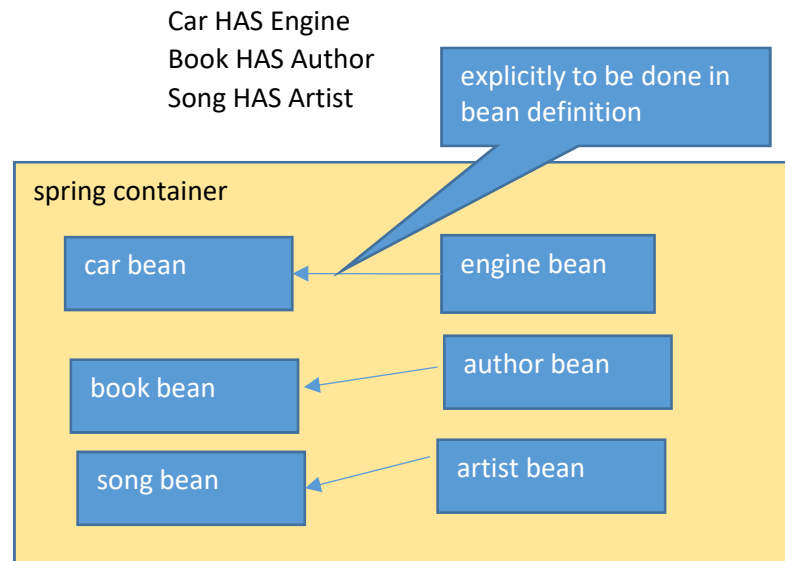
Wiring

Associating / Attaching / Injecting dependency object into dependent object

Manual wiring / Explicit wiring

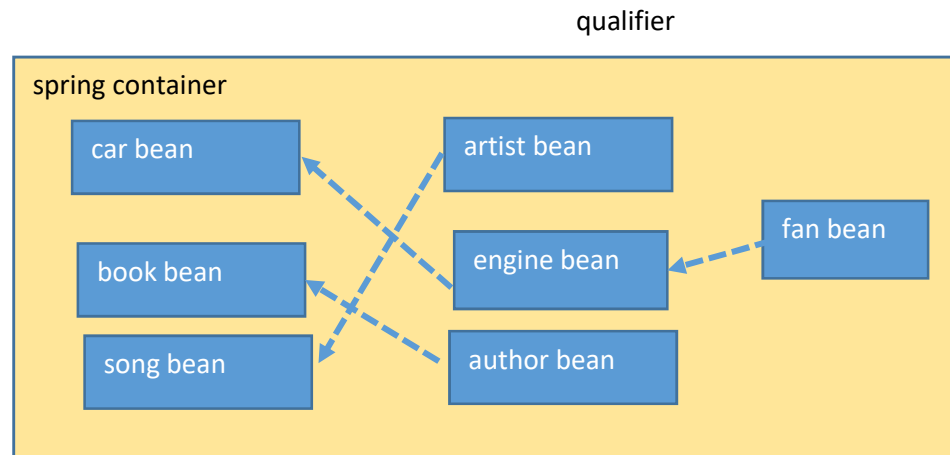
Autowiring / Implicit wiring

scenario1
Manual / Explicit
wiring



Injecting dependency beans manually into dependent bean

scenario 2
Auto wiring
implicit



spring understands which dependent needs which dependency
car bean needs engine bean

spring injects dependency beans into dependent bean automatically

3 types of autowiring

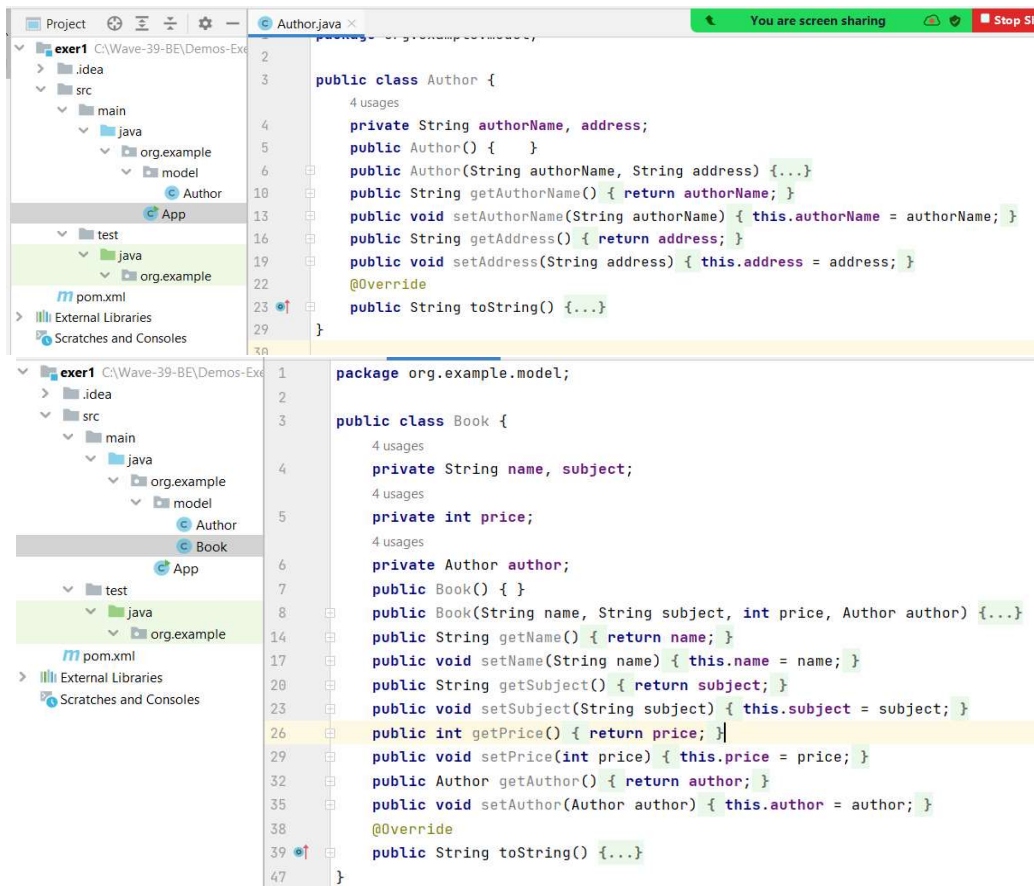
property

setter

constructor *

Create maven application with required model classes

Book HAS Author



The screenshot shows an IDE with a project named 'exer1' located at 'C:\Wave-39-BE\Demos-Ex'. The project structure in the left sidebar includes:

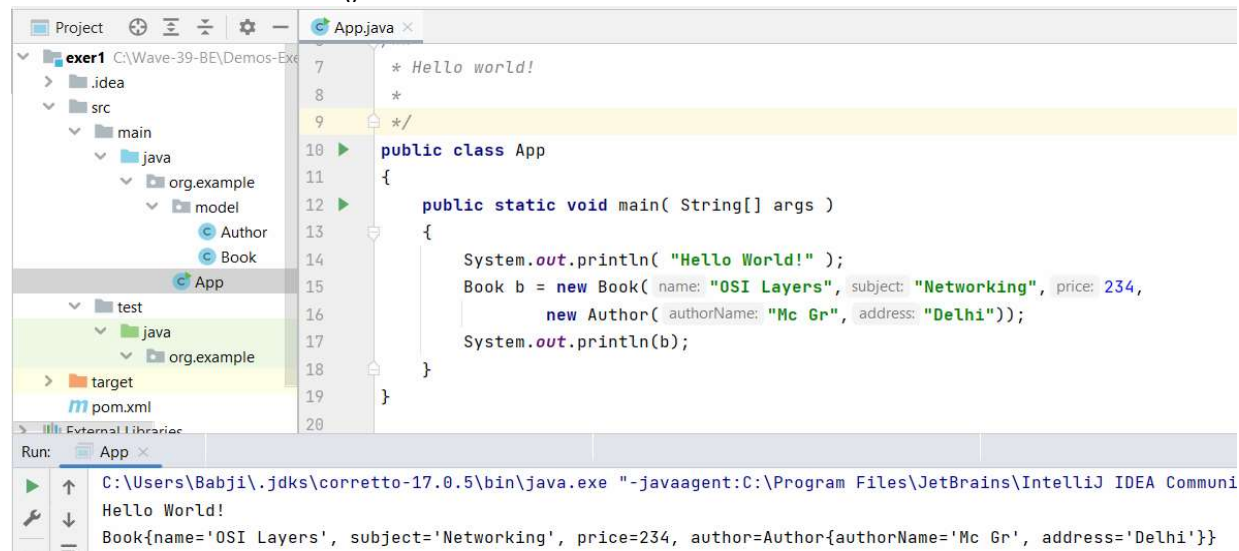
- src
 - main
 - java
 - org.example
 - model
 - Author
 - Book
 - test
 - java
 - org.example
- pom.xml
- External Libraries
- Scratches and Consoles

The main editor displays the code for two classes:

```
2 public class Author {
3     4 usages
4     private String authorName, address;
5     public Author() { }
6     public Author(String authorName, String address) {...}
7     public String getAuthorName() { return authorName; }
8     public void setAuthorName(String authorName) { this.authorName = authorName; }
9     public String getAddress() { return address; }
10    public void setAddress(String address) { this.address = address; }
11    @Override
12    public String toString() {...}
13 }

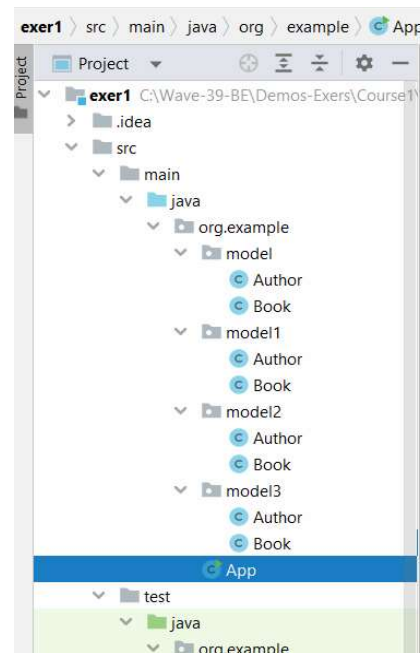
1 package org.example.model;
2
3 public class Book {
4     4 usages
5     private String name, subject;
6     4 usages
7     private int price;
8     4 usages
9     private Author author;
10    public Book() { }
11    public Book(String name, String subject, int price, Author author) {...}
12    public String getName() { return name; }
13    public void setName(String name) { this.name = name; }
14    public String getSubject() { return subject; }
15    public void setSubject(String subject) { this.subject = subject; }
16    public int getPrice() { return price; }
17    public void setPrice(int price) { this.price = price; }
18    public Author getAuthor() { return author; }
19    public void setAuthor(Author author) { this.author = author; }
20    @Override
21    public String toString() {...}
22 }
```

check model classes in main()



Take 3 copies of model classes

- copy1 property autowiring
- copy2 setter autowiring
- copy3 constructor autowiring



Demo 0 Normal way of bean creation and loading beans into container using model package

step 1 add required dependency to pom
spring-context

step 2 make sure Book, Author classes ready in model package

```
graph TD
    org[org.example] --> model[model]
    model --> Author[Author]
    model --> Book[Book]
```

step 3 Create config file and define beans

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure shows a package `org.example` with a sub-package `model` containing `Author` and `Book` classes. The `BookBeanConfig` class is located in a `config` sub-package. The code editor shows the following code:

```
1 package org.example.config;
2
3 import org.example.model.Author;
4 import org.example.model.Book;
5 import org.springframework.context.annotation.Bean;
6
7 public class BookBeanConfig {
8
9     @Bean("book1")
10    public Book getBookBean(){
11        return new Book( name: "Spring annotations", subject: "Spring", price: 345,
12                        new Author( authorName: "M cooper", address: "Pune"));
13    }
14 }
15
```

step 4 in main()
create container using above config file and load beans

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure shows the `BookMain` class in the `org.example` package. The code editor shows the following code:

```
1 package org.example;
2
3 import org.example.config.BookBeanConfig;
4 import org.example.model.Book;
5 import org.springframework.context.ApplicationContext;
6 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
7
8 public class BookMain {
9     public static void main(String[] args) {
10        ApplicationContext context = new AnnotationConfigApplicationContext(BookBeanConfig.class);
11        Book a = context.getBean( s: "book1", Book.class);
12        System.out.println(a);
13    }
14 }
15
```

```
BookMain1 x
C:\Users\Babji\.jdk\corretto-17.0.5\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Ed
Book{name='Spring annotations', subject='Spring', price=345, author=Author{authorName='M cooper', address='Pune'}}
```

Demo 1 auto wiring dependency object into dependent object

property autowiring

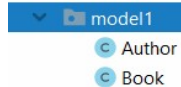
Book, Author classes from model1

config file : BookBeanConfig1

main file : BookMain1

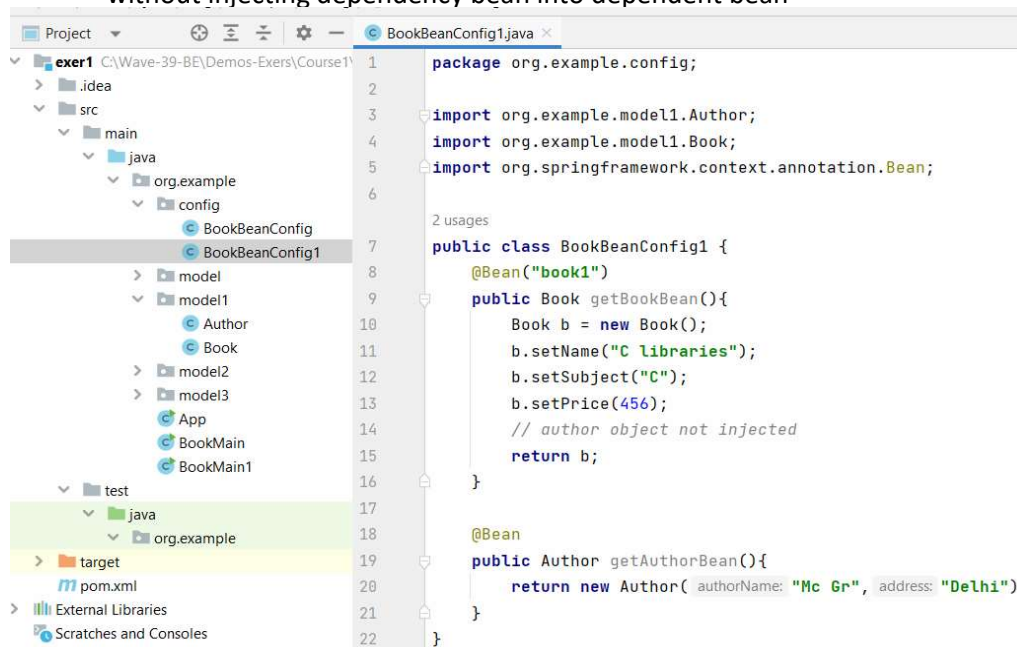
step 1 dependency in pom

step 2 Book, Author classes in model1

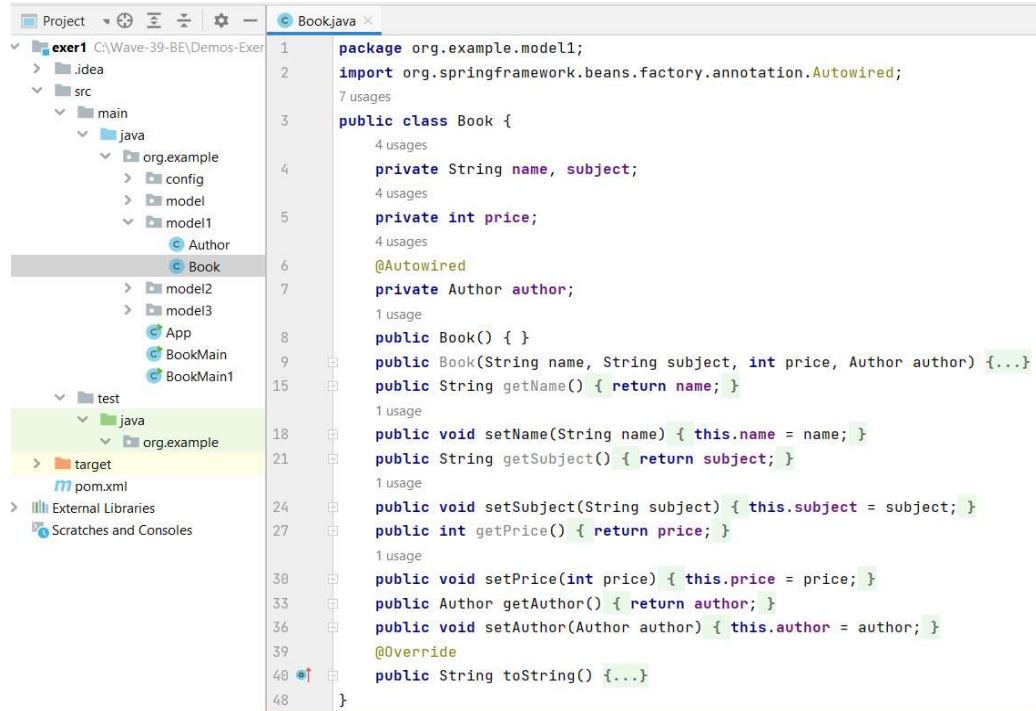


step 3 define beans in configuration file

without injecting dependency bean into dependent bean

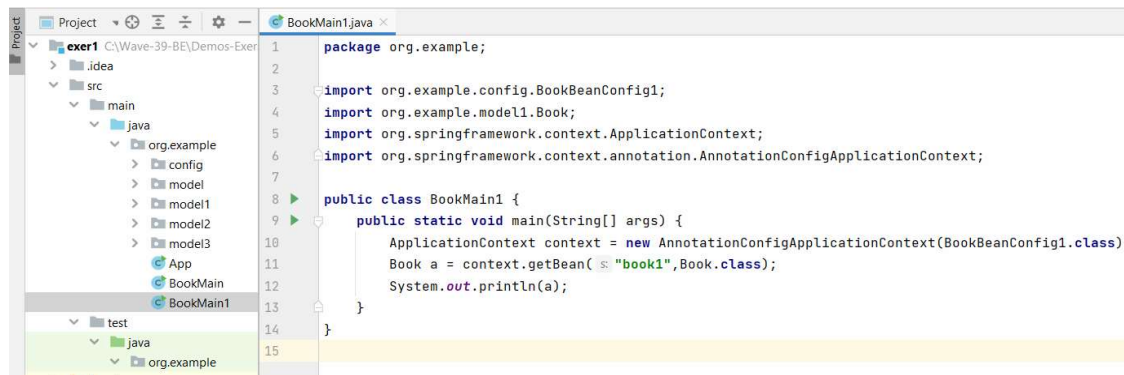


step 4 make author property autowired in book model class



```
1 package org.example.model1;
2 import org.springframework.beans.factory.annotation.Autowired;
3
4 public class Book {
5     private String name, subject;
6     private int price;
7     @Autowired
8     private Author author;
9     public Book() { }
10    public Book(String name, String subject, int price, Author author) {...}
11    public String getName() { return name; }
12    public void setName(String name) { this.name = name; }
13    public String getSubject() { return subject; }
14    public void setSubject(String subject) { this.subject = subject; }
15    public int getPrice() { return price; }
16    public void setPrice(int price) { this.price = price; }
17    public Author getAuthor() { return author; }
18    public void setAuthor(Author author) { this.author = author; }
19    @Override
20    public String toString() {...}
21 }
```

step 5 in main()
create container using above configuration
load beans



```
1 package org.example;
2
3 import org.example.config.BookBeanConfig1;
4 import org.example.model1.Book;
5 import org.springframework.context.ApplicationContext;
6 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
7
8 public class BookMain1 {
9     public static void main(String[] args) {
10         ApplicationContext context = new AnnotationConfigApplicationContext(BookBeanConfig1.class);
11         Book a = context.getBean("book1", Book.class);
12         System.out.println(a);
13     }
14 }
```

```
BookMain1 x
C:\Users\Babji\jdk\corretto-17.0.5\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Book{name='C libraries', subject='C', price=456, author=Author{authorName='Mc Gr', address='Delhi'}}
```

here, author bean is automatically injected into book bean

Demo 2 auto wiring dependency object into dependent object

setter autowiring

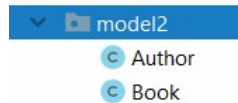
Book, Author classes from model2

config file : BookBeanConfig2

main file : BookMain2

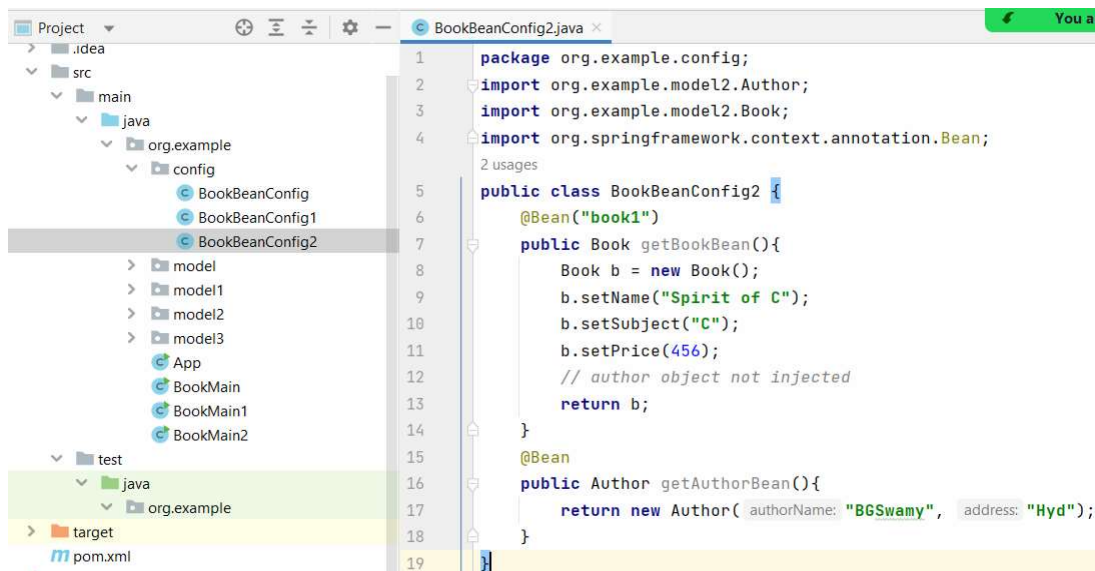
step 1 dependency in pom

step 2 Book, Author classes in model2



step 3 Define beans in configuration file

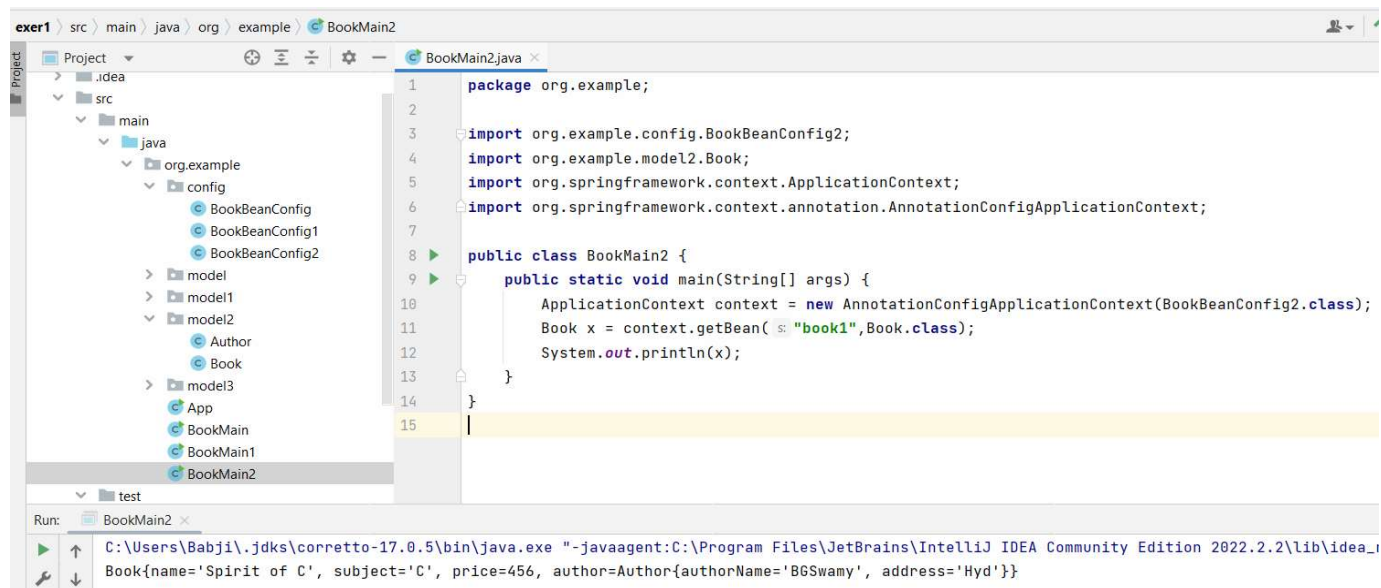
without injecting dependency bean into dependent bean



step 4 modify Book model class in model2 package
 make setAuthor() autowired

```
37           @Autowired
38       public void setAuthor(Author author) {
39           this.author = author;
40       }
```

step 5 in main()
 create container using above configuration file
 load beans

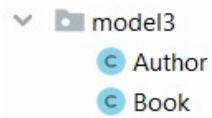


Demo 3 auto wiring dependency object into dependent object
 constructor autowiring

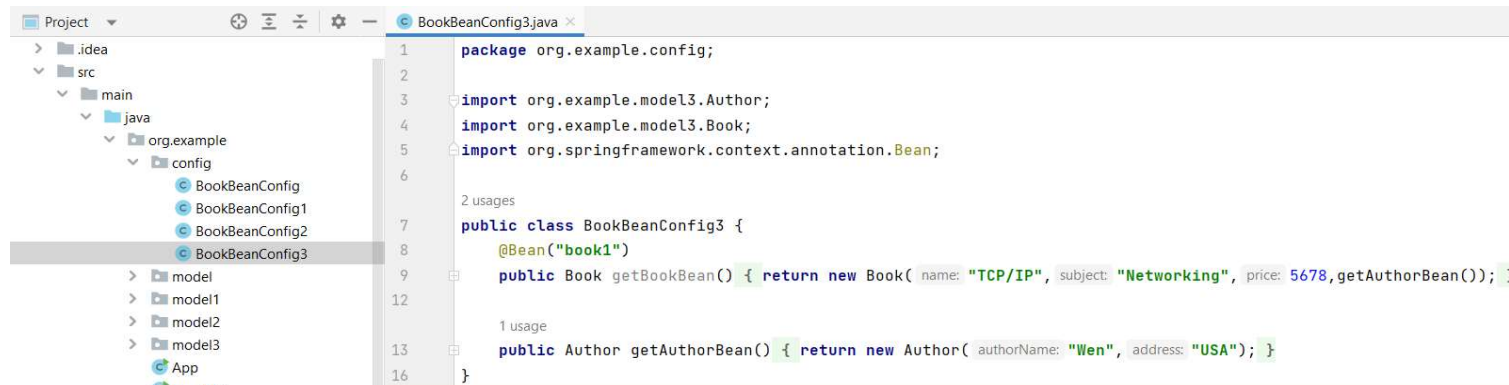
 Book, Author classes from model3
 config file : BookBeanConfig3
 main file : BookMain3

step 1 dependency in pom

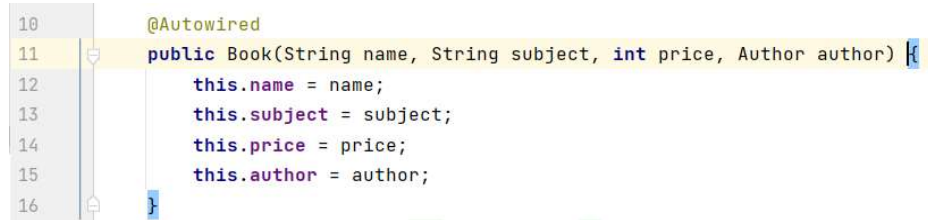
step 2 Book, Author classes in model3



step 3 Define beans in configuration file



step 4 edit Book class and make constructor as autowired



step 5 in main()
create container using above configuration file
load beans

