Contents

[Exercise SDI.2 – Dependency Injection 2](#_Toc25710122)

[A. Change the application in such way that App.java no longer instantiates ProductService but instead retrieves this object from the Spring context. 3](#_Toc25710123)

[B. We want to add an InventoryService object, and inject this InventoryService into the ProductService using Spring dependency injection. 7](#_Toc25710124)

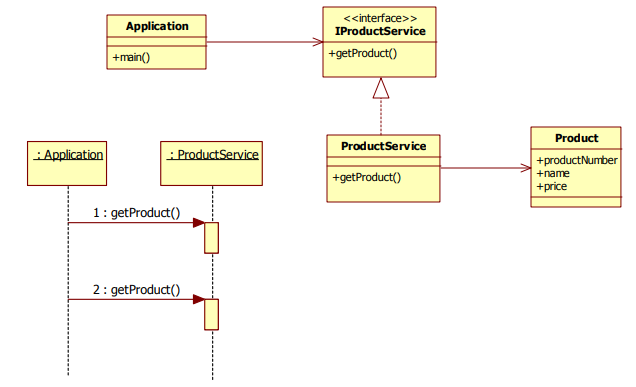
[Exercise SDI.3 – Dependency Injection using Lists 10](#_Toc25710125)

[Problem with current solution 11](#_Toc25710126)

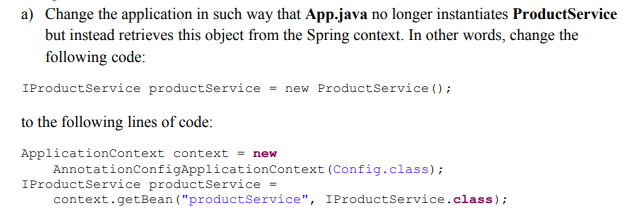
[A. Change the application in such a way that the List is injected into the **BookService** (already filled with IbookSuppliers). In other words, remove the code that instantiates them with new . You will also have to update App.java to retrieve the BookService from Spring. 12](#_Toc25710127)

[B. Add the Borders BookSupplier to the list while only changing configuration (no business related Java code). 15](#_Toc25710128)

## Exercise SDI.2 – Dependency Injection



### Change the application in such way that App.java no longer instantiates ProductService but instead retrieves this object from the Spring context.



#### Solution

For that we need

1. Spring-core dependencies, which means to add dependency in pom.xml
2. Config.class configuration file using notation
3. Use annotations for ProductService class

##### pom.xml: Add spring dependency

<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>cs544</**groupId**>  
 <**artifactId**>W1D2-Dependency\_Injection\_2</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**packaging**>jar</**packaging**>  
  
 <**url**>http://maven.apache.org</**url**>  
  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**maven.compiler.target**>1.8</**maven.compiler.target**>  
 <**maven.compiler.source**>1.8</**maven.compiler.source**>   
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>3.8.1</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-context</**artifactId**>  
 <**version**>5.0.8.RELEASE</**version**>  
 </**dependency**>  
 </**dependencies**>  
</**project**>

##### Config.java

**package** edu.mum.cs544;  
  
**import** org.springframework.context.annotation.ComponentScan;  
**import** org.springframework.context.annotation.Configuration;  
@Configuration  
@ComponentScan(**"edu.mum.cs544"**)  
**public class** Config {  
}

##### ProductService.java

**package** edu.mum.cs544;  
  
**import** org.springframework.stereotype.Component;  
**import** java.util.\*;  
  
@Component  
**public class** ProductService **implements** IProductService {  
 **private** Collection<Product> **productList** = **new** ArrayList<Product>();  
  
 **public** ProductService() {  
 **productList**.add(**new** Product(234, **"LCD TV"**, 895.50));  
 **productList**.add(**new** Product(239, **"DVD player"**, 315.00));  
 **productList**.add(**new** Product(423, **"Plasma TV"**, 992.55));  
 }  
  
 **public** Product getProduct(**int** productNumber) {  
 **for** (Product product : **productList**) {  
 **if** (product.getProductNumber() == productNumber)  
 **return** product;  
 }  
 **return null**;  
 }  
  
}

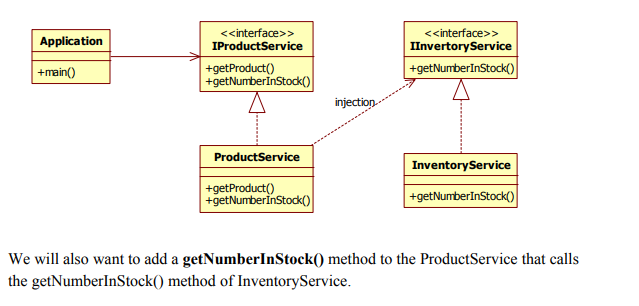
##### AppA.java: change the way of creating productService

**package** edu.mum.cs544;  
  
**import** org.springframework.context.ApplicationContext;  
**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;  
  
**public class** AppA {  
 **public static void** main(String[] args) {  
 *//IProductService productService = new ProductService();* ApplicationContext context = **new** AnnotationConfigApplicationContext(Config.**class**);  
 IProductService productService = context.getBean(**"productService"**, IProductService.**class**);  
  
 Product product1 = productService.getProduct(423);  
 **if** (product1 != **null**) {  
 System.***out***.println(product1.toString());  
 }  
 Product product2 = productService.getProduct(239);  
 **if** (product2 != **null**) {  
 System.***out***.println(product2.toString());  
 }  
 }  
}

#### Output



### We want to add an InventoryService object, and inject this InventoryService into the ProductService using Spring dependency injection.



Configure the Inventory Service to be a spring bean, and make sure that it is properly injected into the ProductService.

#### Solution

##### Dependency injection is done through springbean.xml

*<?***xml version="1.0" encoding="UTF-8"** *?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
http://www.springframework.org/schema/beans/spring-beans.xsd"**>  
 <**bean id="productService" class="edu.mum.cs544.ProductService"**>  
 <**property name="inventoryService" ref="inventoryService"**/>  
 </**bean**>  
 <**bean id="inventoryService" class="edu.mum.cs544.InventoryService"**/>  
</**beans**>

Here inventoryService is injected into ProductService as a property. For that inventoryService must have setter method in ProductService class.

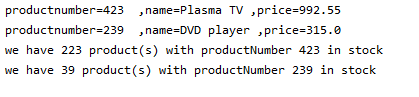
##### ProductService.java

**package** edu.mum.cs544;  
  
**import** org.springframework.stereotype.Component;  
**import** java.util.\*;  
  
@Component  
**public class** ProductService **implements** IProductService {  
 **private** Collection<Product> **productList** = **new** ArrayList<Product>();  
 **private** IInventoryService **inventoryService**;  
 **public** ProductService() {  
 **productList**.add(**new** Product(234, **"LCD TV"**, 895.50));  
 **productList**.add(**new** Product(239, **"DVD player"**, 315.00));  
 **productList**.add(**new** Product(423, **"Plasma TV"**, 992.55));  
 }  
 **public** Product getProduct(**int** productNumber) {  
 **for** (Product product : **productList**) {  
 **if** (product.getProductNumber() == productNumber)  
 **return** product;  
 }  
 **return null**;  
 }  
 @Override  
 **public int** getNumberInStock(**int** productNumber) {  
 **return inventoryService**.getNumberInStock(productNumber);  
 }  
 **public void** setInventoryService(IInventoryService inventoryService) {  
 **this**.**inventoryService** = inventoryService;  
 }  
}

##### AppB.java

**public class** AppB {  
 **public static void** main(String[] args) {  
 ApplicationContext context = **new** ClassPathXmlApplicationContext(**"springbean.xml"**);  
 IProductService productService = context.getBean(**"productService"**, ProductService.**class**);  
  
 Product product1 = productService.getProduct(423);  
 **if** (product1 != **null**) {  
 System.***out***.println(product1.toString());  
 }  
 Product product2 = productService.getProduct(239);  
 **if** (product2 != **null**) {  
 System.***out***.println(product2.toString());  
 }  
 System.***out***.println(**"we have "** + productService.getNumberInStock(423)  
 + **" product(s) with productNumber 423 in stock"**);  
 System.***out***.println(**"we have "** + productService.getNumberInStock(239)  
 + **" product(s) with productNumber 239 in stock"**);  
  
 }  
}

#### Output:



## Exercise SDI.3 – Dependency Injection using Lists

The purpose of this exercise is for you to use the more advanced list configuration feature of dependency injection.

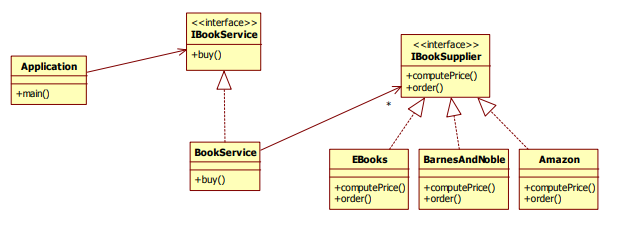
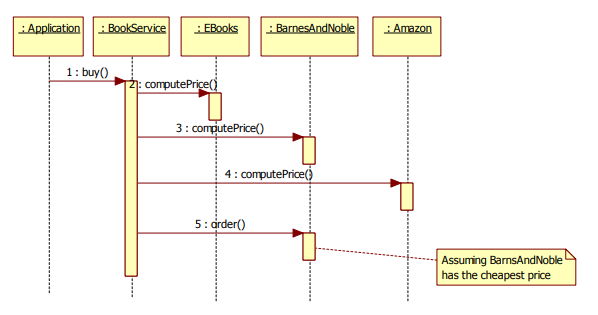


Figure 1Class diagram

The application buys 3 books through the **IBookService** implemented by **BookService**. In the **buy** method, the **BookService** checks each of its **IBookSuppliers**, finding the cheapest one and ordering the book from there.



### Problem with current solution

#### BookService.java hardcoded

The downside of this application is that the BookService is hardcoded with Amazon, EBooks and Barnes & Noble as IBookSuppliers. If we should want to add another book supplier, we would have to go in and change the code.

**package** edu.mum.cs544;  
**import** java.util.\*;  
**public class** BookService **implements** IBookService {  
 **public** List<IBookSupplier> **suppliers** = **new** ArrayList<IBookSupplier>();  
 **public** BookService() {  
 IBookSupplier amazon = **new** Amazon();  
 IBookSupplier barnesandnoble = **new** BarnesAndNoble();  
 IBookSupplier ebooks = **new** EBooks();

BookService is hardcoded with Amazon, EBooks and Barnes & Noble as IBookSuppliers. If we should want to add another book supplier, we would have to go in and change the code.

**suppliers**.add(amazon);  
 **suppliers**.add(barnesandnoble);  
 **suppliers**.add(ebooks);  
 }  
 **public void** buy(Book book) {  
 **double** lowestPrice = 0;  
 IBookSupplier cheapestSupplier = **null**;  
 *// find the cheapest supplier* **for** (IBookSupplier supplier : **suppliers**) {  
 **double** price = supplier.computePrice(book.getIsbn());  
 **if** (cheapestSupplier == **null**) {  
 cheapestSupplier = supplier;  
 lowestPrice = price;  
 } **else** {  
 **if** (price < lowestPrice) {  
 cheapestSupplier = supplier;  
 lowestPrice = price;  
 }  
 }  
 }  
 *// buy with the cheapest supplier* **if** (cheapestSupplier != **null**) {  
 cheapestSupplier.order(book);  
 }  
  
 }  
}

#### App.java

**package** edu.mum.cs544;  
  
**public class** App  
{  
 **public static void** main(String[] args) {  
  
 IBookService bookService = **new** BookService();  
 bookService.buy(**new** Book(**"123433267"**, **"Harry Potter and the Order of the Phoenix"**, **"J.K. Rowling"**));  
 bookService.buy(**new** Book(**"888832678"**, **"Harry Potter and the Sorcerer's Stone"** , **"J.K. Rowling"**));  
 bookService.buy(**new** Book(**"999923156"**, **"Harry Potter and the Goblet of Fire"** ,**"J.K. Rowling"**));  
  
 }  
}

### A. Change the application in such a way that the List is injected into the **BookService** (already filled with IbookSuppliers). In other words, remove the code that instantiates them with new . You will also have to update App.java to retrieve the BookService from Spring.

#### Solution

Remove hardcoded ibookSuppliers and set property using bean property. So that need to define setter method for suppliers.

1. Define springsbean.xml
   1. Inject suppliers
   2. Set suppliers property in springbean.xml
2. Define setter for suppliers in BookService class.
3. Get bookService from spring bean in app.

##### 1.pom.xml: Add spring dependency

<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
   
 <**groupId**>cs544</**groupId**>  
 <**artifactId**>W1D2-Dependency\_Injection\_3</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**packaging**>jar</**packaging**>  
   
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**maven.compiler.target**>1.8</**maven.compiler.target**>  
 <**maven.compiler.source**>1.8</**maven.compiler.source**>   
 </**properties**>  
   
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>3.8.1</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-context</**artifactId**>  
 <**version**>5.0.8.RELEASE</**version**>  
 </**dependency**>  
 </**dependencies**>  
</**project**>

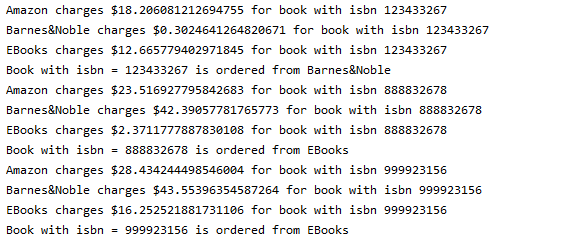
##### Springbean.xml : Define springbean.xml and define bookService bean with suppliers property

*<?***xml version="1.0" encoding="UTF-8"** *?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
http://www.springframework.org/schema/beans/spring-beans.xsd"**>  
 <**bean id="bookService" class="edu.mum.cs544.BookService"**>  
 <**property name="suppliers"**>  
 <**list**>  
 <**ref bean="amazon"** />  
 <**ref bean="barnesAndNoble"** />  
 <**ref bean="ebooks"** />  
 </**list**>  
 </**property**>  
 </**bean**>  
 <**bean id="amazon" class="edu.mum.cs544.Amazon"**></**bean**>  
 <**bean id="barnesAndNoble" class="edu.mum.cs544.BarnesAndNoble"**></**bean**>  
 <**bean id="ebooks" class="edu.mum.cs544.EBooks"**></**bean**>  
</**beans**>

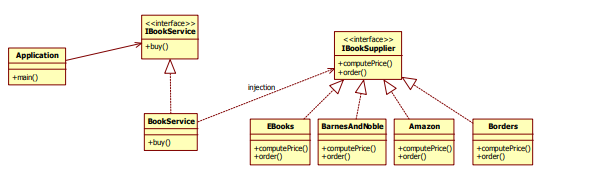
##### App.java : Get bookService from Application context bean in App class.

**package** edu.mum.cs544;  
  
**import** org.springframework.context.ApplicationContext;  
**import** org.springframework.context.support.ClassPathXmlApplicationContext;  
  
**public class** App  
{  
 **public static void** main(String[] args) {  
 ApplicationContext context = **new** ClassPathXmlApplicationContext(**"springbean.xml"**);  
 IBookService bookService = context.getBean(**"bookService"**, BookService.**class**);  
 bookService.buy(**new** Book(**"123433267"**, **"Harry Potter and the Order of the Phoenix"**, **"J.K. Rowling"**));  
 bookService.buy(**new** Book(**"888832678"**, **"Harry Potter and the Sorcerer's Stone"** , **"J.K. Rowling"**));  
 bookService.buy(**new** Book(**"999923156"**, **"Harry Potter and the Goblet of Fire"** ,**"J.K. Rowling"**));  
 }  
}

#### Output



### B. Add the Borders BookSupplier to the list while only changing configuration (no business related Java code).



#### Solution

##### Borders.java: Add the class by implementing IbookSupplier

**package** edu.mum.cs544;  
**public class** Borders **implements** IBookSupplier {  
 @Override  
 **public double** computePrice(String isbn) {  
 **double** price = Math.*random*() \* 45;  
 System.***out***.println(**"EBooks charges $"** + price + **" for book with isbn "** + isbn);  
 **return** price;  
 }  
 @Override  
 **public void** order(Book book) {  
 System.***out***.println(**"Book with isbn = "** + book.getIsbn()  
 + **" is ordered from Borders"**);  
 }  
}

##### Springbean.xml: Add Borders to bean in configuration

*<?***xml version="1.0" encoding="UTF-8"** *?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
http://www.springframework.org/schema/beans/spring-beans.xsd"**>  
 <**bean id="bookService" class="edu.mum.cs544.BookService"**>  
 <**property name="suppliers"**>  
 <**list**>  
 <**ref bean="amazon"** />  
 <**ref bean="barnesAndNoble"** />  
 <**ref bean="ebooks"** />  
 <**ref bean="borders"** />  
 </**list**>  
 </**property**>  
 </**bean**>  
 <**bean id="amazon" class="edu.mum.cs544.Amazon"**></**bean**>  
 <**bean id="barnesAndNoble" class="edu.mum.cs544.BarnesAndNoble"**></**bean**>  
 <**bean id="ebooks" class="edu.mum.cs544.EBooks"**></**bean**>  
 <**bean id="borders" class="edu.mum.cs544.Borders"**></**bean**>  
</**beans**>

#### Output

