Spring Core and Maven

1)Configuring a Basic Spring application

Code:

Created a new Maven project in Eclipse IDE. Inside pom.xml added the following dependencies:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

Inside src/main/java created two packages com.example.repository and com.example.service

Inside com.example.repository created BookRepository.java

**package** com.example.repository;

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("Book saved: "+title);

}

}

Inside com.example.service created BookService.java

**package** com.example.service;

**import** com.example.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

//Setter for spring injecton

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository=bookRepository;

}

**public** **void** addBook(String title) {

System.***out***.println("Adding book: "+title);

bookRepository.saveBook(title);

}

}

Created a folder src/main/resources and added applicationContext.xml   
  
<?xmlversion="1.0" encoding="UTF-8"?>

<beansxmlns="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">

<!-- Repository Bean -->

<beanid="bookRepository"class="com.example.repository.BookRepository"/>

<!-- Service Bean with dependency injection -->

<beanid="bookService"class="com.example.service.BookService">

<propertyname="bookRepository"ref="bookRepository"/>

</bean>

</beans>

Then right click on the folder Build->Use as Source folder.

After that in Eclipse IDE went to Window->Preferences-> Ticked the Download external resources like DTD,XSD.  
  
Then in App.java wrote the following code:

**package** com.example.LibraryManagement;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.example.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context= **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService service= (BookService) context.getBean("bookService");

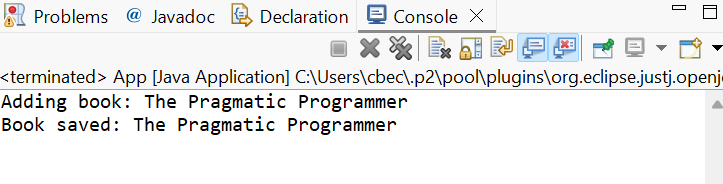
service.addBook("The Pragmatic Programmer");

}

}

Run this App.java as Java application. The following output came.

Output:



2) Implementing Dependency Injection

Code:

In the same project as the previous LibraryManagement, the pom.xml remains the same. Also the applicationContext.xml remains the same. Only a slight change in the BookService class, the BookRepository class and the App.java

BookRepository.java

**package** com.example.repository;

**public** **class** BookRepository{

**public** **void** save(String bookName) {

System.***out***.println("Saving book: "+bookName);

}

}

BookService.java

**package** com.example.service;

**import** com.example.repository.BookRepository;

**public** **class** BookService{

**private** BookRepository bookRepository;

//Setter for Dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository=bookRepository;

}

**public** **void** issueBook(String bookName) {

System.***out***.println("Issuing book: "+bookName);

bookRepository.save(bookName);

}

}

App.java

**package** com.example.LibraryManagement;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.example.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context= **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService= context.getBean("bookService", BookService.**class**);

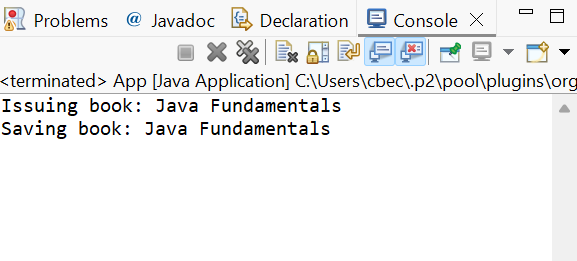
bookService.issueBook("Java Fundamentals");

}

}

On running the App.java as java application, the following output appeared.

Output:



4) Creating and configuring a Maven project

Code:

Created a new Maven project in Eclipse IDE. Inside pom.xml added the following dependencies:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.31</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.31</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.31</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

Also for Java version 1.8 the following had to be done in pom.xml

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

Inside src/main/java created two packages com.example.repository and com.example.service

Inside com.example.repository created BookRepository.java

BookRepository.java

**package** com.example.repository;

**public** **class** BookRepository {

**public** **void** saveBook(String bookName) {

System.***out***.println("Book saved: "+bookName);

}

}

Inside com.example.service created BookService.java

**package** com.example.service;

**import** com.example.repository.BookRepository;

**public** **class**BookService {

**private** BookRepository bookRepository;

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String bookName) {

bookRepository.saveBook(bookName);

}

}

Created a folder src/main/resources and added applicationContext.xml

<?xmlversion="1.0" encoding="UTF-8"?>

<beansxmlns="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">

<beanid="bookRepository"class="com.example.repository.BookRepository"/>

<beanid="bookService"class="com.example.service.BookService">

<!--Setter injection-->

<propertyname="bookRepository"ref="bookRepository"/>

</bean>

</beans>

Then right click on the folder Build->Use as Source folder.

After that in Eclipse IDE went to Window->Preferences-> Ticked the Download external resources like DTD,XSD.

Then in App.java wrote the following code:

**package** com.example.LibraryManagement1;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.example.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context= **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService= (BookService) context.getBean("bookService");

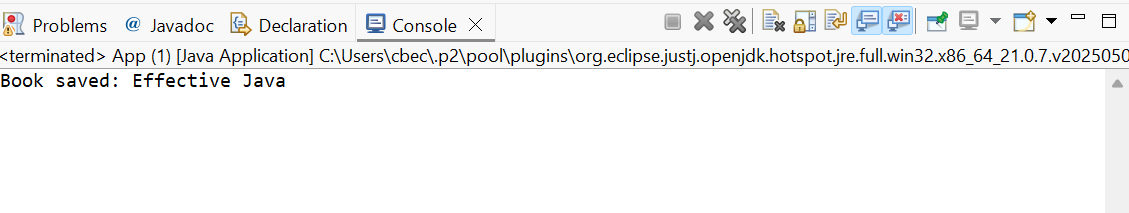
bookService.addBook("Effective Java");

}

}

On running App.java as Java application, the following output was generated:

Output:



7) Implementing Constructor and Setter Injection

Code:

From the previous project structure nothing changed only two changes done. In BookService class a constructor is added and in applicationContext.xml constructor injection is added.

BookService.java

**package** com.example.service;

**import** com.example.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

**private** String libraryName;

//Constructor for constructor injection

**public** BookService(String libraryName) {

**this**.libraryName=libraryName;

}

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository= bookRepository;

}

**public** **void** addBook(String bookName) {

System.***out***.println("Library: "+ libraryName);

System.***out***.println("Adding book: "+ bookName);

bookRepository.saveBook(bookName);

}

}

In applicationContext.xml file

<?xmlversion="1.0" encoding="UTF-8"?>

<beansxmlns="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">

<beanid="bookRepository"class="com.example.repository.BookRepository"/>

<beanid="bookService"class="com.example.service.BookService">

<!-- Constructor injection -->

<constructor-argvalue="City Library"/>

<!--Setter injection-->

<propertyname="bookRepository"ref="bookRepository"/>

</bean>

</beans>

Now on running the same App.java as in the previous exercise, the following ouput is produced

Output:

