1.Configuring a Basic Spring Application Scenario:

Set Up a Spring Project:

```
<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>com.library</groupId>
<artifactId>LibraryManagement</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
  <dependency>
     <groupId>org.springframework</groupId>
     <artifactId>spring-context</artifactId>
     <version>5.3.33</version>
  </dependency>
 </dependencies>
 </project>
```

Configure the Application Context:

```
    <?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/schem
```

```
a/beans
http://www.springframework.org/schema/beans/spring-
beans.xsd">
<bean id="bookRepository"
class="com.library.repository.BookRepository"/>
<!-- BookService Bean -->
<bean id="bookService"
class="com.library.service.BookService">
<property name="bookRepository" ref="bookRepository"/>
</bean>
</beans>
```

Define Service and Repository Classes:

```
    package com.library.repository;
        public class BookRepository {
           public void saveBook(String bookName) {
           System.out.println("Book \"" + bookName + "\"
                saved to the database.");
          }
     }
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
   private BookRepository bookRepository;
```

```
    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService =
context.getBean("bookService", BookService.class);
        bookService.addBook("Spring in Action");
    }
}
```

OUTPUT:

```
Adding book: Spring in Action
Book "Spring in Action" saved to the database.
```

2.Implementing Dependency Injection Scenario:

Modify the XML 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/schem
a/beans
http://www.springframework.org/schema/beans/spring-
beans.xsd"> <bean id="bookRepository"
class="com.library.repository.BookRepository"/> <bean
id="bookService" class="com.library.service.BookService">
<property name="bookRepository" ref="bookRepository"/>
 </bean>
 </beans>
```

Update the BookService Class:

```
  package com.library.service;
  import com.library.repository.BookRepository;
  public class BookService {
  private BookRepository bookRepository;
```

```
   public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }
   public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

Test the Configuration:

```
    package com.library;
    import com.library.service.BookService;
    import org.springframework.context.ApplicationContext;
    import
org.springframework.context.support.ClassPathXmlApplicati
onContext;
    public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");


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

```
        bookService.addBook("Effective Java");
    }
}
```

OUTPUT:

```
Adding book: Effective Java
Book "Effective Java" saved to the database.
```

4: Creating and Configuring a Maven Project Scenario:

Create a New Maven Project:

```
    <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>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
```

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.33</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>5.3.33</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.33</version>
    </dependency>
</dependencies>
```

Configure Maven Compiler Plugin:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
```

```
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

5.Configuring the Spring IoC Container Scenario:

Create Spring Configuration File:

```
        <?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/schem
a/beans
```

```
http://www.springframework.org/schema/beans/spring-
beans.xsd">
   <bean id="bookRepository"
class="com.library.repository.BookRepository"/>
   <bean id="bookService"
class="com.library.service.BookService">
    <property name="bookRepository"
ref="bookRepository"/>
    </bean>
    </beans>
```

Update the BookService Class:

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String title) {
    System.out.println("BookService: Adding book - " + title);
      bookRepository.saveBook(title);
    }
}
```

BookRepository Class (for completeness):

```
package com.library.repository;
public class BookRepository {
    public void saveBook(String title) {
        System.out.println("BookRepository: Saving book - " +
title);
    }
}
package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService =
context.getBean("bookService", BookService.class);
        bookService.addBook("Clean Code");
    }
}
```

OUTPUT:

```
BookService: Adding book - Clean Code
BookRepository: Saving book - Clean Code
```

7:Implementing Constructor and Setter Injection
Scenario:

Configure Constructor Injection applicationContext.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"
     <bean id="bookRepository"
class="com.library.repository.BookRepository"/>
     <bean id="bookService"
class="com.library.service.BookService">
      <constructor-arg value="Library Service Alpha"/>
        <property name="bookRepository"
ref="bookRepository"/>
   </bean>
   </beans>
```

Update the BookService Class:

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
   private BookRepository bookRepository;
   private String serviceName;
```

```
    public BookService(String serviceName) {
        this.serviceName = serviceName;
    }
    public void setBookRepository(BookRepository
bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String title) {
        System.out.println(serviceName + ": Adding book - " +
title);
        bookRepository.saveBook(title);
    }
}

package com.library.repository;
public class BookRepository {
    public void saveBook(String title) {
        System.out.println("BookRepository: Saving book - " +
title);
    }
}

package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = context.getBean("bookService", BookService.class);
        bookService.addBook("Java Concurrency in Practice");
    }
}
```

OUTPUT:

```
Library Service Alpha: Adding book - Java Concurrency in Practice
BookRepository: Saving book - Java Concurrency in Practice
```

9:Creating a Spring Boot Application Scenario:

Create a Spring Boot Project:

```
   <dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Book.java:

```
package com.library.model;
import jakarta.persistence.*;
@Entity
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
```

```
   private String author;
   public Long getId() { return id; }
   public void setId(Long id) { this.id = id; }
   public String getTitle() { return title; }
   public void setTitle(String title) { this.title = title; }
   public String getAuthor() { return author; }
   public void setAuthor(String author) { this.author = author;
}
}
```

BookRepository.java:

```
   package com.library.repository;
   import com.library.model.Book;
   import
org.springframework.data.jpa.repository.JpaRepository;
  public interface BookRepository extends
  JpaRepository<Book, Long> {
}
```

```
package com.library.controller;
import com.library.model.Book;
import com.library.repository.BookRepository;
import
org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
@RestController
@RequestMapping("/books")
public class BookController {
    @Autowired
    private BookRepository bookRepository;
    @GetMapping
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }
    @GetMapping("/{id}")
    public Optional<Book> getBookById(@PathVariable Long
id) {
        return bookRepository.findById(id);
    }
    @PostMapping
    public Book createBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }
    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id,
@RequestBody Book book) {
        book.setId(id);
        return bookRepository.save(book);
    }
    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        bookRepository.deleteById(id);
    }
}

package com.library;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicati
on;
@SpringBootApplication
public class LibraryManagementApplication {
 public static void main(String[] args) {
 SpringApplication.run(LibraryManagementApplication.class,
args);
    }
}
```

OUTPUT: