# Spring Core and Maven Exercises

Exercise 1: Configuring a Basic Spring Application  
  
Objective:  
Set up a simple Spring application with basic bean configuration.  
  
Steps:  
  
1. Create a Java Project  
 - Use your preferred IDE (like Eclipse or IntelliJ).  
 - Set up a standard Maven project structure.  
  
2. Add Spring Core Dependency in pom.xml:  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
</dependencies>  
  
3. Create Bean Classes:  
  
// HelloWorld.java  
public class HelloWorld {  
 public void sayHello() {  
 System.out.println("Hello from Spring!");  
 }  
}  
  
4. Spring Configuration File (beans.xml):  
  
<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="helloWorld" class="HelloWorld" />  
</beans>  
  
5. Main Application:  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class App {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");  
 HelloWorld obj = (HelloWorld) context.getBean("helloWorld");  
 obj.sayHello();  
 }  
}  
  
  
Exercise 2: Implementing Dependency Injection  
  
Objective:  
Understand constructor and setter-based dependency injection.  
  
Steps:  
  
1. Create Dependent Classes:  
  
public class Address {  
 private String city;  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public void display() {  
 System.out.println("City: " + city);  
 }  
}  
  
public class Employee {  
 private Address address;  
  
 // Constructor-based DI  
 public Employee(Address address) {  
 this.address = address;  
 }  
  
 public void showDetails() {  
 System.out.println("Employee Details:");  
 address.display();  
 }  
}  
  
2. Update beans.xml:  
  
<bean id="address" class="Address">  
 <property name="city" value="Chennai"/>  
</bean>  
  
<bean id="employee" class="Employee">  
 <constructor-arg ref="address"/>  
</bean>  
  
3. Main Application:  
  
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");  
Employee emp = (Employee) context.getBean("employee");  
emp.showDetails();  
  
  
Exercise 4: Creating and Configuring a Maven Project  
  
Objective:  
Set up a full Maven-based Spring application with dependencies.  
  
Steps:  
  
1. Create Maven Project:  
 - Use mvn archetype:generate or create via IDE.  
  
2. Add Dependencies to pom.xml:  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
</dependencies>  
  
3. Follow the same class and XML setup as in Exercise 1 or 2.  
  
4. Build and Run the Project:  
 - Use mvn clean install to build.  
 - Run via IDE or java -cp target/classes your.package.App