* At its core, Spring offers a container, often referred to as the Spring application context, that creates and manages application components.
* The act of wiring beans together is based on a pattern known as dependency injection (DI).
* In recent versions of Spring, however, a Java-based configuration is more common. The following Java-based configuration class is equivalent to the XML configuration:

@Configuration

public class ServiceConfiguration {

@Bean

public InventoryService inventoryService()

{ return new InventoryService(); }

@Bean

public ProductService productService()

{

return new ProductService(inventoryService());

}

}

The @Configuration annotation indicates to Spring that this is a configuration class that will provide beans to the Spring application context. The configuration’s class methods are annotated with @Bean, indicating that the objects they return should be added as beans in the application context (where, by default, their respective bean IDs will be the same as the names of the methods that define them).