# 1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

# 2) What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

#### 1) Predefined Templates

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

#### 2) Loose Coupling

The Spring applications are loosely coupled because of dependency injection.

#### 3) Easy to test

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

#### 4) Lightweight

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

#### 5) Fast Development

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

#### 6) Powerful abstraction

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

#### 7) Declarative support

It provides declarative support for caching, validation, transactions and formatting.

# 3) What are the benefits of Spring Framework?

Following are the benefits of Spring framework:

* Extensive usage of Components
* Reusability
* Decoupling
* Reduces coding effort by using pattern implementations such as singleton, factory, service locator etc.
* Removal of leaking connections
* Declarative transaction management
* Easy to integrate with third party tools and technologies.

# 4)What are the differences between EJB and Spring ?

   Spring and EJB feature comparison.TransactionManagement, Declarative transaction support, Persistence, Declarative security, Distributed computing.

|  |  |  |
| --- | --- | --- |
| **Feature** | **EJB** | **Spring** |
| Transaction management | * Must use a JTA transaction manager. * Supports transactions that span remote method calls. | * Supports multiple transaction environments through its PlatformTransactionManager interface, including JTA, Hibernate, JDO, and JDBC. * Does not natively support distributed transactions—it must be used with a JTA transaction manager. |
| Declarative transaction support | * Can define transactions declaratively through the deployment descriptor. * Can define transaction behavior per method or per class by using the wildcard character \*. * Cannot declaratively define rollback behavior—this must be done programmatically. | * Can define transactions declaratively through the Spring configuration file or through class metadata. * Can define which methods to apply transaction behavior explicitly or by using regular expressions. * Can declaratively define rollback behavior per method and per exception type. |
| Persistence | Supports programmatic bean-managed persistence and declarative container managed persistence. | Provides a framework for integrating with several persistence technologies, including JDBC, Hibernate, JDO, and iBATIS. |
| Declarative security | * Supports declarative security through users and roles. The management and implementation of users and roles is container specific. * Declarative security is configured in the deployment descriptor. | * No security implementation out-of-the box. * Acegi, an open source security framework built on top of Spring, provides declarative security through the Spring configuration file or class metadata. |
| Distributed computing | Provides container-managed remote method calls. | Provides proxying for remote calls via RMI, JAX-RPC, and web services. |

Spring 4 features:

# 5) Here are the some important improvements in Spring 4.x

* Removed Deprecated Packages and Methods, [check out the API Differences Report](http://docs.spring.io/spring-framework/docs/3.2.4.RELEASE_to_4.0.0.RELEASE/)
* Java 8 Support[
* JSR-335 [Lambda expressions](https://javabeat.net/2012/05/a-sneak-peak-at-the-lambda-expressions-in-java-8/)
* JSR-310 Date-Time value types for Spring data binding and formatting.
* Java EE 6 and 7 or above is now considered the baseline for Spring Framework 4

Java EE 7 Enterprise Concurrency support.

* Groovy Bean Definition DSL, [read more about this API](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/groovy/GroovyBeanDefinitionReader.html).
* Core Container Improvements
* General Web Improvements

WebSocket, SockJS, and STOMP Messaging[ STOMP (Simple (or Streaming) Text-Oriented Messaging Protocol) is a simple text-oriented protocol, similar to HTTP].

* [JSR-343](http://jcp.org/en/jsr/detail?id=343) JMS 2.0. latest version
* [JSR-338](http://jcp.org/en/jsr/detail?id=338) JPA 2.1. latest version
* Testing Improvements

**And Spring 3.x which introduced lot of new features like:**

* Spring MVC Test Framework
* Asynchronous MVC processing on Servlet 3.0
* custom @Bean definition annotations in @Configuration classes
* @Autowired and @Value to be used as meta-annotations
* Concurrency refinements across the framework
* loading WebApplicationContexts in the TestContext framework
* JCache 0.5 (JSR-107)

**1. Removed Deprecated Packages and Methods**

All deprecated packages, and many deprecated classes and methods have been removed with version 4.0. If you are upgrading from a previous release of Spring, you should ensure that you have fixed any deprecated calls that you were making to outdated APIs. This should be the challenging point for those who want to migrate their old spring applications to the new version. As a caution, if you are running the old spring application, then first thing to get rid of your deprecated APIs before migrating to the latest version.

For a complete set of changes, [check out the API Differences Report](http://docs.spring.io/spring-framework/docs/3.2.4.RELEASE_to_4.0.0.RELEASE/).

**2. Java 8 Support**

Spring Framework 4.0 provides support for several Java 8 features. You can make use of lambda expressions and method references with Spring’s callback interfaces. There is first-class support for java.time (JSR-310), and several existing annotations have been retrofitted as @Repeatable. You can also use Java 8’s parameter name discovery (based on the -parameters compiler flag) as an alternative to compiling your code with debug information enabled.

**Some of the Java SE 8 features to be supported include:**

* JSR-335 Lambda expressions.
* JSR-310 Date-Time value types for Spring data binding and formatting.
* Support for the new 1.8 byte code format (required to support Lambda expressions).

Spring 4.0 has increased the minimum recommendation to Java 6.0. Also it states that any new project recommended to use the Java 7.0 for their projects with Spring 4.0. It still supports the lower version Java 6 and 7 without any problem.

**3. Java EE 6 and 7**

Java EE version 6 or above is now considered the baseline for Spring Framework 4, with the JPA 2.0 and Servlet 3.0 specifications being of particular relevance. It is possible to run your application in Servlet 2.5, but it is recommended to use Servlet 3.0 environment.

**4. Groovy Bean Definition DSL**

With Spring Framework 4.0 it is now possible to define external bean configuration using a Groovy DSL. [Read more about this API](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/groovy/GroovyBeanDefinitionReader.html).

**5. Core Container Improvements**

There have been several general improvements to the core container:

* Spring now treats [generic types as a form ofqualifier](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-generics-as-qualifiers) when injecting Beans. For example, if you are using a Spring Data Repository you can now easily inject a specific implementation: @Autowired Repository<Customer> customerRepository.
* If you use Spring’s meta-annotation support, you can now develop custom annotations that [expose specific attributes from the source annotation](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-meta-annotations).
* Beans can now be Ordered when they are [autowired intolists and arrays](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-autowired-annotation). Both the @Ordered annotation and Ordered interface are supported.
* The @Lazy annotation can now be used on injection points, as well as @Bean definitions.
* The [@Description annotation has been added](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-java-bean-description) for developers using Java-based configuration.
* A generalized model for [conditionally filtering beans](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-java-conditional) has been added via the @Conditional annotation. This is similar to @Profile but allows for user-defined strategies to be developed.
* [CGLIB-based proxy classes](http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#aop-pfb-proxy-types) no longer require a default constructor. Support is provided via the [objenesis](http://code.google.com/p/objenesis/) library which is repackaged *inline* and distributed as part of the Spring Framework. With this strategy, no constructor at all is being invoked for proxy instances anymore.
* There is managed time zone support across the framework now, e.g. on LocaleContext.

**6. General Web Improvements**

Deployment to Servlet 2.5 servers remains an option, but Spring Framework 4.0 is now focused primarily on Servlet 3.0+ environments. If you are using the Spring MVC Test Framework you will need to ensure that a Servlet 3.0 compatible JAR is in your test classpath.

In addition to the WebSocket support mentioned later, the following general improvements have been made to Spring’s Web modules:

* You can use the new ***@RestController*** annotation with Spring MVC applications, removing the need to add ***@ResponseBody*** to each of your ***@RequestMapping*** methods.
* The ***AsyncRestTemplate***class has been added, allowing non-blocking asynchronous support when developing REST clients.
* Spring now offers comprehensive timezone support when developing Spring MVC applications.

**7. WebSocket, SockJS, and STOMP Messaging**

A new spring-websocket module provides comprehensive support for WebSocket-based, two-way communication between client and server in web applications. It is compatible with JSR-356, the Java WebSocket API, and in addition provides SockJS-based fallback options (i.e. WebSocket emulation) for use in browsers that don’t yet support the WebSocket protocol (e.g. Internet Explorer < 10).

A new spring-messaging module adds support for STOMP as the ***WebSocket***sub-protocol to use in applications along with an annotation programming model for routing and processing STOMP messages from ***WebSocket***clients. As a result an ***@Controller*** can now contain both ***@RequestMapping*** and ***@MessageMapping*** methods for handling HTTP requests and messages from WebSocket-connected clients. The new spring-messaging module also contains key abstractions from the Spring Integration project such as ***Message, MessageChannel, MessageHandler***, and others to serve as a foundation for messaging-based applications.

**8. Testing Improvements**

In addition to pruning of deprecated code within the spring-test module, Spring Framework 4.0 introduces several new features for use in unit and integration testing.

* Almost all annotations in the spring-test module (e.g., *@ContextConfiguration, @WebAppConfiguration, @ContextHierarchy, @ActiveProfiles, etc.*) can now be used as meta-annotations to create custom composed annotations and reduce configuration duplication across a test suite.
* Active bean definition profiles can now be resolved programmatically, simply by implementing a custom ***ActiveProfilesResolver***and registering it via the resolver attribute of *@ActiveProfiles*.
* A new ***SocketUtils***class has been introduced in the spring-core module which enables you to scan for free TCP and UDP server ports on localhost. This functionality is not specific to testing but can prove very useful when writing integration tests that require the use of sockets, for example tests that start an in-memory SMTP server, FTP server, Servlet container, etc.
* As of Spring 4.0, the set of mocks in the *org.springframework.mock.web* package is now based on the Servlet 3.0 API. Furthermore, several of the Servlet API mocks (e.g., **MockHttpServletRequest**, **MockServletContext**, etc.) have been updated with minor enhancements and improved configurability.