On 12-December-2013, Spring IO has officially announced that Spring Framework 4.0 GA is available for the production use. Version 4.0 is the latest major release of the Spring Framework and the first to fully support Java 8 features.

Here I summarize the prominent features.

New Features and Enhancements in Spring Framework 4.0

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from a previous release of Spring, you should ensure that you have fixed any deprecated calls that you were making to outdated APIs.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.

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.

5. Core Container Improvements
There have been several general improvements to the core container:

- Spring now treats generic types as a form ofqualifier 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.
- Beans can now be Ordered when they are autowired intolists and arrays. 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 for developers using Java-based configuration.
- A generalized model for conditionally filtering beans 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 no longer require a default constructor. Support is provided via the 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.