Some Points related to Spring Framework

================================

* Spring is modular by design.

1. Main projects by Spring to support infrastructure needs of a project:-

<https://spring.io/projects>

1. **Spring Boot --** Takes an opinionated view of building Spring applications and gets you up and running as quick as possible.
2. **Spring Framework --** Provides core support for dependency injection, transaction management, web apps, data access, messaging and more.
3. **Spring Cloud Data Flow --** An orchestration service for composable data microservice applications on modern runtimes.
4. **Spring Cloud --** Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.
5. **Spring Data --** Provides a consistent approach to data access – relational, non-relational, map-reduce and beyond.
6. **Spring Integration --** Supports the well-known Enterprise Integration Patterns vial lightweight messaging and declarative adapters.
7. **Spring Batch --**
8. **Spring Security --** Protects your application with comprehensive and extensible authentication and authorization support.
9. Spring Hateoas
10. Spring Rest Docs
11. Spring AMQP
12. Spring Mobile
13. Spring For Android
14. **Spring Web Flow --** Supports building web applications with controlled navigation such as checking in for a flight or applying for a loan.
15. **Spring Web Services --** Facilitates the development of contract-first soap web services.
16. Spring LDAP
17. **Spring Session –** Spring session provides an API and implementations for managing a user’s session information.
18. Spring Shell
19. Spring Flo
20. Spring Kafka
21. Spring State Machine
22. **Spring IO Platform –** Provides a cohesive, versioned platform for building modern applications. It is a modular, enterprise-grade distribution that delivers a curated set of dependencies.

* Developers are constantly challenged with choosing the most effective runtime, programming model, and architecture for their application's requirements and team's skill set.
* some [use cases](https://spring.io/blog/2016/06/07/notes-on-reactive-programming-part-i-the-reactive-landscape#reactive-use-cases) are best handled by a technology stack based on **synchronous blocking I/O architecture,** whereas others would be better served by an **asynchronous non-blocking stack** built on the reactive design principles [Reactive Streams Specification](http://www.reactive-streams.org/).
* Spring Framework 5 provides a new reactive web stack called Spring WebFlux, which is offered side by side with the traditional Spring MVC web stack.
* **Reactive Stack -** Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next generation processors and handle massive numbers of concurrent connections.
* **Servlet Stack -** Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.

Terminologies used in Spring

=======================

* Bean
* Bean Definition
* Scope of Bean
* Nature of the Bean
* Container
* Inversion of Control (IOC)
* Dependency Injection (DI)
* Aspect Oriented Programming (AOP)
* Configuration Metadata (XML based, Annotation based, Java based)
* Constuctor Injection
* Setter Injection
* AutoWiring (@AutoWired / @Inject)
* Limitations and Disadvantages of AutoWiring.
* Method Injection (<https://spring.io/blog/2004/08/06/method-injection/>)
* Web-Aware ApplicationContext
* @PostConstruct and @PreDestroy
* BeanPostProcessor
* LifeCycleProcessor
* ApplicationContextAware and BeanNameAware
* Other Aware Interfaces
* Child Bean definition
* Integration Interfaces / BeanPostProcessor
* BeanFactoryPostProcessor
* PropertySourcesPlaceholderConfigurer
* PropertyOverrideConfigurer
* FacotryBean
* @Autowired / @Inject / @Value / @Resource / @primary /@Qualifier
* @Required /@Resource
* @Order / @Bean / @primary /@DependsOn / @Nullable