1. [What is Dependency Injection?](http://www.javainterview.in/p/spring-interview-questions.html#question1)  
   Dependency Injection (DI) je koncept u softverskom inženjeringu koji se koristi za upravljanje zavisnostima između objekata. Osnovna ideja iza Dependency Injection-a je da objekti ne bi trebalo da budu odgovorni za kreiranje ili pronalaženje svojih zavisnosti, već da te zavisnosti budu ubrizgane (injektovane) u objekte iz spoljašnjeg izvora.

Umesto da objekti sami instanciraju ili pronalaze svoje zavisnosti, DI omogućava da te zavisnosti budu obezbeđene izvan objekta. To se obično postiže kroz konstruktor ili metode objekta koje se nazivaju injekcija zavisnosti. Kada se objekat kreira, DI kontejner (eng. DI container) ili drugi mehanizam za DI će automatski obezbediti potrebne zavisnosti.

1. [Why is Spring one of the most popular Java related frameworks?](http://www.javainterview.in/p/spring-interview-questions.html#question2)  
   Spring je jedan od najpopularnijih Java okvira iz više razloga. Prvo, on omogućava lako upravljanje zavisnostima i olakšava implementaciju Dependency Injectiona. Takođe, Spring pruža bogat set funkcionalnosti za razvoj web aplikacija, podržava različite slojeve aplikacije (npr. prezentacioni sloj, poslovni sloj, sloj za pristup podacima) i pruža integraciju sa različitim tehnologijama i framework-ima. Spring takođe podržava aspekte kao što su sigurnost, transakcije i upravljanje greškama. Sve ove karakteristike čine Spring popularnim izborom za razvoj Java aplikacija.
2. [What are the different modules in Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html#question3)  
   Spring Framework se sastoji od nekoliko modula, koji pružaju podršku za različite aspekte razvoja aplikacija. Neki od tih modula su:

* Spring Core: Spring Core modul je osnova celokupnog Spring Framework-a. Pruža osnovne koncepte kao što su Inverzija kontrole (IoC) i Ubacivanje zavisnosti (Dependency Injection - DI). Takođe sadrži i implementaciju kontejnera za ubacivanje zavisnosti (Dependency Injection Container) koji omogućava upravljanje životnim ciklusom objekata.
* Spring MVC: Spring MVC (Model-View-Controller) modul pruža podršku za razvoj web aplikacija zasnovanih na modelu MVC. Ovaj modul omogućava definisanje kontrolera, pregleda (view) i modela, kao i obradu HTTP zahteva i generisanje HTTP odgovora. Spring MVC takođe podržava različite tehnologije za prezentaciju, uključujući JSP, Thymeleaf i druge.
* Spring Data: Spring Data modul pruža podršku za rad sa različitim izvorima podataka, kao što su relacijske baze podataka, NoSQL baze podataka, mapiranje objekata i druge tehnologije perzistencije podataka. Ovaj modul pojednostavljuje upotrebu i interakciju sa različitim izvorima podataka kroz zajednički API i podržava različite ORM (Object-Relational Mapping) alate kao što su Hibernate, JPA i drugi.
* Spring Security: Spring Security modul pruža funkcionalnosti za implementaciju bezbednosti u Spring aplikacijama. Omogućava autentifikaciju, autorizaciju, zaštitu od napada i druge bezbednosne mehanizme. Ovaj modul se može koristiti za zaštitu web aplikacija, REST servisa i ostalih delova sistema.

1. [Can you give an overview of a web application that is implemented using Spring related Modules?](http://www.javainterview.in/p/spring-interview-questions.html#question4)  
   Web aplikacija implementirana korišćenjem Spring-related modula obično ima sledeće komponente:

* Kontroleri (Controllers): Ovi kontroleri primaju HTTP zahteve, obrađuju ih i izvršavaju odgovarajuće akcije. Oni komuniciraju sa servisnim slojem aplikacije.
* Servisi (Services): Servisi sadrže poslovnu logiku aplikacije i obavljaju specifične zadatke. Oni koriste repozitorijume za pristup podacima.
* Repozitorijumi (Repositories): Repozitorijumi pružaju sloj za pristup podacima. Oni komuniciraju sa bazom podataka ili drugim izvorima podataka.
* Pogledi (Views): Pogledi su odgovorni za prikazivanje podataka korisniku. Mogu biti predstavljeni kroz HTML, JSON, XML ili druge formate.

1. [What is the simplest way of ensuring that we are using single version of all Spring related dependencies?](http://www.javainterview.in/p/spring-interview-questions.html#question5)  
   Najjednostavniji način za osiguranje korišćenja jedne verzije svih Spring-related zavisnosti je korišćenje upravljača zavisnostima (dependency management) alata kao što je Maven ili Gradle. Ovi alati omogućavaju definisanje zajedničke verzije za sve Spring-related zavisnosti u projektu.
2. [What are the major features in different versions of Spring ?](http://www.javainterview.in/p/spring-interview-questions.html#question6)  
   Različite verzije Spring Frameworka dolaze sa različitim karakteristikama i poboljšanjima. Evo nekoliko primera:

* Spring 3: Uvođenje Java konfiguracije (Java-based configuration), podrška za RESTful web servise, unapređenje aspektno-orijentisanog programiranja (AOP).
* Spring 4: Uvođenje HTML5 podrške, podrška za Java 8, unapređenje Spring MVC frameworka, podrška za WebSocket komunikaciju.
* Spring 5: Uvođenje podrške za reaktivno programiranje (Reactive Programming), poboljšanja performansi i podrška za Kotlin programski jezik.

1. [What are the latest specifications supported by Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html#question7)  
   Spring 4.0 podržava sledeće specifikacije:

* Servlet 3.0: Omogućava korišćenje naprednih funkcionalnosti servlet kontejnera.
* JavaServer Faces (JSF) 2.2: Pruža podršku za razvoj web aplikacija koristeći JSF tehnologiju.
* Java Persistence API (JPA) 2.1: Omogućava persistenciju objekata u bazu podataka kroz objektno-relaciono mapiranje (ORM).

1. [Can you describe some of the new features in Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html#question8)  
   U Springu 4.0 uvedene su neke nove funkcionalnosti, kao što su:

* Podrška za Java 8: Spring 4.0 podržava nove jezičke konstrukcije u Java 8, kao što su lambda izrazi, metode zadužene za funkcionalno programiranje, nova Java 8 vremena i datum API.
* Unapređenja u Spring MVC: Spring 4.0 donosi mnogo poboljšanja u Spring MVC frameworku, uključujući podršku za RESTful web servise, unapređenu podršku za WebSocket komunikaciju, uklanjanje nekih deprecated API-ja, i mnoge druge funkcionalnosti za razvoj web aplikacija.
* Obrada izuzetaka: Spring 4.0 donosi poboljšanja u mehanizmu obrade izuzetaka, uključujući novi način rukovanja izuzecima kroz anotacije i podršku za fleksibilno mapiranje izuzetaka na HTTP statusne kodove.

1. [What is auto-wiring?](http://www.javainterview.in/p/spring-interview-questions.html#question9)  
   Automatsko povezivanje (autowiring) u Springu je mehanizam koji omogućava automatsko rezoluciju zavisnosti između objekata. Umesto da eksplicitno konfigurišemo sve zavisnosti, Spring može automatski otkriti odgovarajuće objekte iz kontejnera i ubaciti ih u odgovarajuće tačke u kodu. Ovo olakšava razvoj i održavanje aplikacija, jer se smanjuje potreba za ručnim konfigurisanjem i ubacivanjem zavisnosti. Automatsko povezivanje se može postaviti na nivou klase, metode ili polja, koristeći odgovarajuće anotacije ili XML konfiguraciju.
2. [What would happen in Spring Container finds multiple bean definitions matching the property to be auto wired?](http://www.javainterview.in/p/spring-interview-questions.html#question10)  
   Ako Spring kontejner pronađe više definicija bean-a koje odgovaraju svojstvu koje se automatski povezuje, doći će do greške i neće biti moguće izvršiti automatsko povezivanje. U takvim slučajevima, potrebno je precizno specificirati koja definicija bean-a treba da se koristi koristeći dodatne anotacije ili konfiguraciju. Alternativno, možemo koristiti kvalifikatore (qualifiers) kako bismo precizno odabrali odgovarajuću definiciju bean-a za automatsko povezivanje.
3. [How is Spring’s singleton bean different from Gang of Four Singleton Pattern?](http://www.javainterview.in/p/spring-interview-questions.html#question11)  
   Singleton bean u Springu se odnosi na objekat koji je konfigurisan kao singleton unutar Spring kontejnera. To znači da će kontejner kreirati samo jednu instancu tog objekta i koristiti je za sve zahteve za taj bean. S druge strane, Singleton obrazac iz Gang of Four se odnosi na dizajn obrasca koji definiše klasu koja može imati samo jednu instancu u celoj aplikaciji. Ove dve koncepcije su slične u smislu da obe podržavaju samo jednu instancu objekta, ali Singleton bean u Springu je specifičan za Spring kontejner i njegovo upravljanje objektima.
4. [How do you represent stateful bean in Spring?](http://www.javainterview.in/p/spring-interview-questions.html#question12)  
   U Springu, stanoviti bean (stateful bean) se predstavlja kroz prototype scope. Prototype scope podrazumeva da se za svaki zahtev za takav bean kreira nova instanca, umesto da se koristi jedinstvena instanca. Ovo omogućava da svaka instanca ima svoje stanje i da ne deli stanje sa drugim instancama. Kada se koristi stanoviti bean, svaki put kada se zatraži takav bean, kontejner će kreirati novu instancu sa svezim stanjem.
5. [How do you use values defined in a property file in an application context xml?](http://www.javainterview.in/p/spring-interview-questions.html#question13)  
   Da biste koristili vrednosti definisane u property fajlu u XML kontekstu aplikacije, možete koristiti "PropertyPlaceholderConfigurer" bean koji se konfiguriše u XML fajlu. Ovaj bean omogućava čitanje vrednosti iz property fajla i injektovanje tih vrednosti u odgovarajuće elemente u XML konfiguraciji. Takođe, možete koristiti izraze za vrednosti (value expressions) ili SpEL (Spring Expression Language) za dinamičko izračunavanje vrednosti na osnovu property fajla.
6. [How is validation done using Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html#question14)  
   U Spring Frameworku, validacija se vrši pomoću "Validator" interfejsa i "DataBinder" objekta. "Validator" interfejs definiše metodu za proveru ispravnosti objekta, dok "DataBinder" objekat se koristi za povezivanje podataka sa objektom i vršenje validacije. Validacija se obično vrši kroz anotacije na model objektima, gde se definišu pravila validacije za svako polje. Prilikom vezivanja podataka, Spring će automatski primenjivati validaciju na osnovu definisanih pravila.
7. [How do you implement cross cutting concerns in a web application?](http://www.javainterview.in/p/spring-interview-questions.html#question15)  
   Presecanje (cross-cutting concerns) u web aplikaciji se može implementirati pomoću aspektno orijentisanog programiranja (AOP) u Spring Frameworku. Pomoću AOP-a možete izdvojiti presecajući funkcionalnosti kao odvojene aspekte i primeniti ih na odgovarajućim tačkama u kodu, bez uplitanja u osnovnu logiku aplikacije. Na primer, možete koristiti aspekt za logovanje, transakcije, sigurnost ili praćenje performansi. Spring pruža podršku za AOP putem Spring AOP modula ili AspectJ frameworka, koji se mogu integrisati u Spring aplikaciju.
8. [What is an Aspect and Pointcut in AOP?](http://www.javainterview.in/p/spring-interview-questions.html#question16)  
   Aspekt (Aspect) u AOP-u predstavlja presecajuću funkcionalnost koja se primenjuje na određene tačke u kodu. To može biti funkcionalnost kao što je logovanje, transakcije, sigurnost ili bilo koja druga funkcionalnost koja preseca osnovnu logiku aplikacije. Pointcut definiše određene tačke u kodu gde će se aspekt primenjivati. Na primer, pointcut može biti definisan kao određena metoda ili grupa metoda koje će biti presecajuće tačke za određeni aspekt.
9. [What are the different types of AOP advices?](http://www.javainterview.in/p/spring-interview-questions.html#question17)  
   Različite vrste AOP saveta su:

* Before advice: Izvršava se pre izvršavanja određene tačke u kodu.
* After returning advice: Izvršava se nakon uspešnog izvršavanja određene tačke u kodu.
* After throwing advice: Izvršava se nakon što se dogodi izuzetak prilikom izvršavanja određene tačke u kodu.
* After advice: Izvršava se nakon izvršavanja određene tačke u kodu, bez obzira na to da li je izvršavanje bilo uspešno ili je došlo do izuzetka.
* Around advice: Obuhvata određenu tačku u kodu i omogućava izvršavanje dodatne logike pre, posle ili umesto te tačke u kodu.

1. [How do you define transaction management for Spring – Hibernate integration?](http://www.javainterview.in/p/spring-interview-questions.html#question18)  
   Upravljanje transakcijama za integraciju Springa i Hibernatea se može postići kroz deklarativno ili programsko upravljanje. Deklarativno upravljanje se postiže kroz konfiguraciju Spring transakcionog menadžera i anotacija nad odgovarajućim metodama ili klasama koje zahtevaju transakcije. Programsko upravljanje se postiže kroz Springovu "PlatformTransactionManager" API, gde možete ručno započeti, potvrditi ili poništiti transakciju.
2. [How do you choose the framework to implement AOP - Spring AOP or AspectJ?](http://www.javainterview.in/p/spring-interview-questions.html#question19)  
   Izbor između Spring AOP-a i AspectJ-a za implementaciju AOP-a zavisi od potreba i zahteva projekta. Spring AOP je jednostavniji i lakši za korišćenje, ali podržava samo ograničen skup AOP funkcionalnosti. AspectJ, s druge strane, je moćan alat za AOP sa bogatim setom funkcionalnosti, ali zahteva dodatnu konfiguraciju i moguće je da zahteva integraciju sa build procesom. Ako vam je potrebna osnovna AOP podrška i jednostavnost integracije, možete koristiti Spring AOP. Ako vam je potrebna napredna AOP funkcionalnost, kao što je presecanje na nivou klase ili veća fleksibilnost, možete razmotriti korišćenje AspectJ-a.
3. [What are the different mock objects provided by Spring test framework?](http://www.javainterview.in/p/spring-interview-questions.html#question20)  
   Spring testni okvir pruža različite mock objekte za testiranje, uključujući:

* MockMvc: Za testiranje Spring MVC kontrolera.
* MockRestServiceServer: Za testiranje REST servisa i HTTP klijenta.
* MockDataSource: Za testiranje sloja podataka i pristup bazi podataka.
* MockJmsTemplate: Za testiranje JMS poruka i redova.
* MockEnvironment: Za simuliranje okruženja tokom testiranja.

1. [What are the utility methods available to test JDBC classes?](http://www.javainterview.in/p/spring-interview-questions.html#question21)  
   Spring testni okvir pruža nekoliko korisnih metoda za testiranje JDBC klasa, uključujući:

* JdbcTemplate: Omogućava izvršavanje SQL upita i manipulaciju podacima u testnom okruženju.
* EmbeddedDatabase: Omogućava kreiranje ugrađene baze podataka za testiranje.
* TransactionTemplate: Omogućava upravljanje transakcijama prilikom testiranja.

1. [How do you setup a Session Factory to integrate Spring and Hibernate?](http://www.javainterview.in/p/spring-interview-questions.html#question22)  
   Za integraciju Springa i Hibernatea, možete konfigurisati Session Factory putem Springovog "LocalSessionFactoryBean" objekta. Ovaj objekat se konfiguriše u XML fajlu i definiše Hibernate konfiguraciju, kao i druge relevantne parametre kao što su izvori podataka i prečice za mapiranje. Session Factory se može koristiti za kreiranje Hibernate Session objekata, koje možete koristiti za izvršavanje Hibernate upita i manipulaciju podacima.
2. [How do you implement caching with Spring framework?](http://www.javainterview.in/p/spring-interview-questions.html#question23)  
   Implementacija keširanja sa Spring Frameworkom uključuje konfiguraciju keširanja na nivou aplikacije i upotrebu keširanih objekata. Možete konfigurisati keširanje putem Spring Cache anotacija nad odgovarajućim metodama ili klasama koje zahtevaju keširanje. Možete konfigurisati keš provajder (npr. EhCache, Redis) i njegove parametre u Spring XML konfiguraciji. Kada se metoda poziva, Spring će proveriti keširanu vrednost i, ako postoji, vratiti je umesto da se izvrši stvarni poziv metode.
3. [What are the important features of Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question31)  
   Neke važne funkcionalnosti Spring Batcha su:

* Paralelno izvršavanje i raspodela posla: Spring Batch omogućava paralelno izvršavanje poslova, što poboljšava performanse i smanjuje vreme izvršavanja. Takođe omogućava raspodelu posla na više čvorova za distribuirano izvršavanje.
* Skalabilnost: Spring Batch se može skalirati kako bi se nosio sa velikim obimima podataka i visokom opterećenju.
* Transakcionalnost: Spring Batch obezbeđuje transakcionalnu obradu podataka i automatski upravlja potvrdama i poništenjima transakcija.
* Oporavak od grešaka: Spring Batch ima ugrađenu podršku za oporavak od grešaka, što omogućava ponovno izvršavanje neuspešnih poslova i manipulaciju greškama u obradi podataka.
* Konfigurabilnost: Spring Batch pruža fleksibilnu konfiguraciju poslova putem XML-a ili anotacija, što olakšava prilagođavanje poslova prema specifičnim potrebama.
* Praćenje i nadgledanje: Spring Batch pruža mehanizme za praćenje i nadgledanje izvršavanja poslova, uključuj

1. [What are the important concepts related to setting up a Job in Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question32)  
   Važni koncepti vezani za podešavanje posla u Spring Batchu uključuju:

* Job: Definiše celokupni posao koji se izvršava, uključujući korake, tokove, slaganje, strategije oporavka od grešaka i druge parametre.
* Step: Predstavlja jedan korak u poslu, koji obično obuhvata obradu određenog dela podataka. Može sadržavati ItemReader, ItemProcessor i ItemWriter.
* ItemReader: Komponenta koja čita podatke iz izvora, na primer baze podataka ili datoteka.
* ItemProcessor: Komponenta koja prima podatke iz ItemReadera i vrši neku vrstu transformacije ili obrade nad njima pre nego što se proslede ItemWriteru.
* ItemWriter: Komponenta koja piše podatke u odredište, na primer bazu podataka ili datoteke.

1. [What are the different ItemReader and ItemWriter implementations available with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question33)  
   Spring Batch pruža nekoliko implementacija ItemReadera i ItemWritera koje se mogu koristiti za različite potrebe. Nekoliko primeraka uključuje JdbcCursorItemReader, JpaPagingItemReader, FlatFileItemReader kao implementacije ItemReadera, i JdbcBatchItemWriter, JpaItemWriter, FlatFileItemWriter kao implementacije ItemWritera.
2. [How do you start running a Spring Batch Job?](http://www.javainterview.in/p/spring-interview-questions.html#question34)  
   Za pokretanje izvršavanja Spring Batch posla, potrebno je koristiti JobLauncher interfejs koji se može konfigurisati putem Spring konteksta. JobLauncheru se prosleđuje referenca na definisani posao koji želite pokrenuti, a zatim se poziva metoda launchJob(). Posao će se pokrenuti i izvršiti prema definisanim koracima i konfiguraciji.
3. [How do you configure parallel execution of steps with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question35)  
   Da biste omogućili paralelno izvršavanje koraka u Spring Batchu, možete koristiti konfiguraciju Chunk-oriented obrade podataka. Ova konfiguracija omogućava podešavanje veličine chunkova, tj. broja podataka koji se obrađuju u jednoj iteraciji. Podešavanjem atributa task-executor u konfiguraciji koraka možete odabrati executor koji će izvršavati korake paralelno, na primer ThreadPoolTaskExecutor. Na taj način, više koraka se može izvršavati istovremeno, poboljšavajući performanse i brzinu izvršavanja.