1. در مورد SOLID می‌دانید؟

بله، یک مجموعه SOLID از پنج اصل اساسی در طراحی نرم افزار است که به کمک آن ها می توان توسعه داد، با کیفیت بالا و منعطف ایجاد کرد. این پنج اصل عبارتند از:

اصل مسئولیت واحد (SRP): یک کلاس باید فقط یک مسئولیت داشته باشد و باید به طور کامل آن مسئولیت را بر عهده داشته باشد. این اصل برای کاهش پیچیدگی و افزایش قابل قبولی و قابلیت نگهداری کد استفاده از قرار می‌گیرد.

اصل باز/بسته (OCP): کلاس‌ها باید برای تغییر در آینده باز باشند، اما برای تغییر در حال حاضر بسته هستند. به این معنی که کلاس‌ها باید تغییرات را در آینده داشته باشند، اما به شکلی طراحی می‌شوند که در حال حاضر نیازی به تغییر وجود نداشته باشد.

Liskov Substitution Principle (LSP): اصلاح یک کلاس توسط یک زیر کلاس نباید برنامه را تغییر دهد. به این معنی که اگر کلاسی از کلاسی دیگر به عنوان پدر استفاده شود، باید جایگزینی با آن باشد.

اصل تفکیک واسط (ISP): باید ارتباط‌ها را به چند مورد کوچکتر و ویژه‌تر تقسیم کنند، به جای استفاده از یک ارتباط بزرگ. به این صورت که هر کلاس فقط به چیزی نیاز دارد که وابسته باشد.

Dependency Inversion Principle (DIP): برای کاهش وابستگی، کد باید به تابعیت برنامه‌نویسی اعمال شود. کد باید به اشیاء وابسته باشد، بلکه باید به ابسته‌ترین لایه‌ها (مانند ارتباط‌ها) وابسته باشد.

1. در مورد Design Patterns می‌دانید؟

بله، Design Patterns به عنوان یک الگوی معماری در نرم‌افزارها استفاده می‌شوند. این الگوها به صورت تجربی، معمولا توسط توسعه‌دهندگان مختلف شناخته شده‌اند و در بسیاری از مسائل طراحی و توسعه نرم‌افزارها کاربرد دارند.

برخی از پرکاربردترین Design Patterns عبارتند از:

Factory Method Pattern: این الگو به توسعه دهندگان اجازه می‌دهد تا یک شیء را بر اساس ورودی ساخته و بازگرداندن آن به کاربر بپردازند.

Singleton Pattern: این الگو برای تضمین اینکه فقط یک نمونه از یک کلاس ساخته شود و در تمامی بخش‌های برنامه قابل دسترس باشد، استفاده می‌شود.

Observer Pattern: در این الگو، یک شیء، به عنوان فراگیر، به تعدادی شیء، به عنوان ناظر، اعلام می‌کند که وضعیت آن تغییر کرده است و شیء‌های ناظر، در پاسخ به تغییرات، وضعیت خود را به روز می‌کنند.

Strategy Pattern: این الگو به توسعه دهندگان امکان می‌دهد تا یک الگوی رفتاری را از کلاس جدا کنند و آن را به یک شیء دیگر تزریق کنند تا به سادگی با تغییرات آن کار کنند.

Decorator Pattern: در این الگو، تابعیت یک شیء را به شیء‌های دیگری می‌افزاید و اجازه می‌دهد تا کلاس‌ها را بدون تغییر در کد اصلی توسعه دهند.

همچنین الگوهای دیگری مانند Adapter، Proxy، Facade و Command نیز برای کاهش پیچیدگی کد و افزایش انعطاف‌پذیری در نرم‌افزارها مورد استفاده قرار می‌گیرند.

1. اسپرینگ چیست و چه کاربردی دارد؟

Spring یکی از محبوب‌ترین فریم‌ورک‌های جاوا است که برای توسعه برنامه‌های مبتنی بر جاوا استفاده می‌شود. این فریم‌ورک، ابزارها و کتابخانه‌های بسیاری را برای تسهیل توسعه برنامه‌های جاوا فراهم می‌کند و بسیاری از کارهای رایج در توسعه نرم‌افزار را از جمله Dependency Injection، Aspect-Oriented Programming، JDBC Template و Transaction Management را پیاده‌سازی می‌کند.

با استفاده از Spring، توسعه دهندگان می‌توانند برنامه‌های بزرگ و پیچیده‌تری را با تسهیل فرایند توسعه، پیاده‌سازی و تست بسیاری از امکانات و قابلیت‌های Spring ایجاد کنند. به عنوان مثال، از قابلیت Dependency Injection که در Spring پیاده‌سازی شده است، برای جداسازی مسئولیت‌های مختلف کلاس‌ها و ماژول‌ها، بهبود قابلیت تست و کاهش وابستگی‌های بین کدها استفاده می‌شود.

به طور خلاصه، Spring یک فریم‌ورک قدرتمند و مفید است که توسعه دهندگان را قادر می‌سازد تا برنامه‌های پویا، پویایی جسته و پیچیده را با کمترین هزینه و تلاش پیاده‌سازی کنند. با این حال، برای استفاده از Spring، نیاز به آشنایی با مفاهیم مبتنی بر این فریم‌ورک مانند Dependency Injection و Aspect-Oriented Programming و غیره دارید.

1. برخی از ویژگی‌های اسپرینگ چیست؟

برخی از ویژگی‌های اسپرینگ (Spring) عبارت‌اند از:

Dependency Injection (DI): در اسپرینگ از مفهوم Dependency Injection استفاده می‌شود که باعث جداسازی بخش‌های مختلف برنامه و کاهش وابستگی‌ها می‌شود. این مفهوم به عنوان یکی از اصول SOLID نیز شناخته می‌شود.

Aspect-Oriented Programming (AOP): با استفاده از AOP در اسپرینگ، می‌توانید بخش‌های مختلف برنامه را به عنوان وظایف جانبی و جداگانه از برنامه اصلی برنامه نویسی کنید. این به شما کمک می‌کند تا کد برنامه‌ی خود را ساده‌تر و قابل نگهداری‌تر کنید.

Spring MVC: Spring MVC یک فریم‌ورک Web جاوا است که برای توسعه برنامه‌های وب استفاده می‌شود. این فریم‌ورک برای توسعه برنامه‌های با الگوی Model-View-Controller (MVC) طراحی شده است.

Spring Boot: Spring Boot یکی از قدرتمندترین ویژگی‌های اسپرینگ است که برای توسعه برنامه‌های سریع و آسان با استفاده از Spring فریم‌ورک ایجاد شده است. با استفاده از Spring Boot، شما می‌توانید به سرعت یک برنامه Spring توسعه دهید و با کمترین تنظیمات و پیکربندی‌ها، برنامه خود را در کمترین زمان ممکن به دست آورید.

Spring Security: Spring Security یک فریم‌ورک امنیتی برای برنامه‌های وب است که برای مدیریت و کنترل دسترسی‌ها و احراز هویت کاربران استفاده می‌شود.

Spring Data: Spring Data یکی از بخش‌های Spring است که برای فراهم کردن یک API ساده و کارآمد برای ارتباط با پایگاه داده استفاده می‌شود.

1. اسپرینگ چگونه به ما کمک می‌کند تا Dependency Injection را پیاده‌سازی کنیم؟

یکی از ویژگی های اصلی فریمورک Spring، پشتیبانی از Dependency Injection (DI) است. با استفاده از DI، شما می توانید اشیاء را به یکدیگر متصل کنید و از جمله وابستگی هایی که برنامه شما برای اجرا خود دارد را مدیریت کنید.

با استفاده از DI در Spring، شما می توانید به جای ساختن اشیاء خود بصورت دستی، به Spring فریمورک بگویید که اشیاء را به شما بدهد. برای این کار، شما به جای خود Spring Container (یا IoC Container) را به کار می‌برید. این کانتینر مسئول ساختن و مدیریت اشیاء و وابستگی هایشان در برنامه شماست.

شما می توانید با استفاده از annotation هایی مانند @Autowired، @Resource و @Inject به Spring بگویید که چگونه باید اشیاء را متصل کند. همچنین با استفاده از XML Configuration، می توانید اشیاء را به یکدیگر متصل کنید.

با استفاده از DI در Spring، کد شما به صورت قابل تست، قابلیت توسعه و قابلیت استفاده مجدد بهبود می‌یابد. به این صورت که شما نیازی به ساختن اشیاء خود بصورت دستی ندارید و Spring مسئول آن است. همچنین وابستگی هایی که در برنامه شما وجود دارد به صورت خودکار توسط Spring مدیریت می‌شود و شما نیازی به مدیریت آنها ندارید.

1. برخی از Annotation های اسپرینگ چیست؟

Annotation های Spring به شما کمک می کنند تا به کانتینر Spring بگویید که چگونه باید اجزای برنامه شما را تعبیه کند. برخی از Annotation های مهم Spring عبارتند از:

@Component: این Annotation به کلاس ها می گوید که باید به عنوان یک component (مولفه) در Spring کانتینر ثبت شوند.

@Autowired: این Annotation برای تزریق (Injection) وابستگی ها (Dependencies) به کلاس ها و اشیاء Spring استفاده می شود. با استفاده از این Annotation، Spring به صورت خودکار تلاش می کند تا یک bean مناسب را پیدا کند و آن را برای injection به متغیر مورد نظر ارائه دهد.

@Configuration: این Annotation به Spring می گوید که باید از کلاس مورد نظر به عنوان یک مبدل پیکربندی استفاده کند. این کلاس ها معمولا برای تنظیم کردن bean ها و محیط پیکربندی Spring استفاده می شوند.

@Controller: این Annotation به کلاس ها می گوید که باید به عنوان یک controller (کنترل کننده) در Spring MVC شناخته شوند.

@Service: این Annotation به کلاس ها می گوید که باید به عنوان یک service (سرویس) در Spring شناخته شوند.

@Repository: این Annotation به کلاس ها می گوید که باید به عنوان یک repository (مخزن) در Spring شناخته شوند.

این Annotation ها تنها بخشی از Annotation های Spring هستند و در Spring بسیاری از Annotation های دیگری نیز وجود دارند که برای موارد مختلفی مانند تنظیم کردن برخی ویژگی ها و ویژوالیزه کردن کد استفاده می شوند.

1. برای مدیریت transaction ها در اسپرینگ از چه روش‌هایی می‌توانید استفاده کنید؟

در اسپرینگ، برای مدیریت transaction ها، می‌توانید از روش‌های زیر استفاده کنید:

۱. بوسیله TransactionTemplate: با استفاده از این روش، می‌توانید در کد خود، transaction ها را مدیریت کنید. برای این منظور، باید یک نمونه از TransactionTemplate ایجاد کرده و با استفاده از متدهای آن، transaction های مورد نیاز را شروع، commit و یا rollback کنید.

۲. بوسیله @Transactional Annotation: این Annotation در اسپرینگ برای مشخص کردن transaction ها در کلاس‌ها و متدها استفاده می‌شود. با اعمال این Annotation بر روی یک کلاس یا متد، تمامی عملیات‌های این متد در یک transaction اجرا می‌شود. این Annotation همچنین امکان تعیین مشخصات transaction مانند نوع isolation level، propagation و timeout را نیز فراهم می‌کند.

۳. بوسیله Spring AOP: در این روش، با استفاده از AOP، می‌توانید transaction ها را به صورت خودکار به متدها اعمال کنید. برای این منظور، می‌توانید یک Aspect تعریف کنید و با استفاده از Annotation‌های Spring AOP، عملیات‌های transactional را برای آن تعریف کنید. سپس این Aspect را به برنامه اضافه کرده و تمامی متدهایی که باید در transaction اجرا شوند، با استفاده از این Annotation‌ها مشخص کنید.

به طور کلی، برای مدیریت transaction ها در اسپرینگ، روش‌های مختلفی وجود دارد که انتخاب روش مناسب، بستگی به نیازهای خاص پروژه دارد.

1. برخی از ابزارهای مانیتورینگ و لاگینگ موجود در Spring چیست؟

در Spring برای مانیتورینگ و لاگینگ برنامه‌ها، ابزارهای متعددی وجود دارد که برخی از آن‌ها عبارت‌اند از:

Spring Boot Actuator: یکی از مهم‌ترین ابزارهای مانیتورینگ در Spring است که اطلاعات جامعی در مورد وضعیت برنامه و متریک‌های آن را ارائه می‌دهد. به کمک Actuator، می‌توانید اطلاعاتی مانند فضای مصرفی در حافظه، اطلاعات مربوط به Thread، زمان پاسخ‌دهی به درخواست‌ها و... را دریافت کنید.

Spring Boot Admin: یک رابط کاربری برای مانیتورینگ و مدیریت برنامه‌های Spring Boot است. این ابزار اطلاعاتی مانند وضعیت سلامتی برنامه‌ها، اطلاعات مربوط به متریک‌ها و پردازش‌ها و ... را به صورت گرافیکی نشان می‌دهد.

Spring Cloud Sleuth: یک ابزار تراس‌کینگ برای برنامه‌های Spring است که به کمک آن می‌توانید فرایند یک درخواست در سیستم را تعقیب کنید و مشکلات موجود در آن‌ها را پیدا کنید.

Spring AOP: یک ابزار برای پیاده‌سازی مفهوم AOP در برنامه‌های Spring است. با استفاده از AOP، می‌توانید مشخص کنید که بخش‌های مختلف برنامه چگونه تغییر کنند و چگونه مانیتورینگ شوند.

Log4j: یک ابزار لاگینگ برای برنامه‌های Java است که می‌توانید با استفاده از آن لاگ‌های برنامه‌ی خود را به سادگی در فایل‌های مختلف ذخیره کنید و در آینده به راحتی پیدا کنید.