Objedname - Backend test

Bc. Patrik Duch
August 21, 2024

Contents

1 Otázky

1.1 1. Jak vytvoříte nový modul v MestJs? Jaké jsou výhody použití modulů v aplikaci?

NestJS byl inspirován ekosystémem z Angular 2+, kde je možné organizovat kód do zminovaných modulů. Výhoda spočívá v organizaci kódu, znovupoužitelnost a jednodušší správa závislostí. Tahle modulární architektura usnadňuje testování a samotnou údržbu aplikace.

Existují dva ekvivalentní CLI dotazy:

nest generate module module-name

nest g mo module-name

Moduly lze také vytvářet manuálně.

1.2 2. Co jsou middleware v NestJS a jak se liší od middleware v jiných frameworcích? Kdy byste použili middleware v aplikaci?

Middleware v NestJS jsou funkce, které se spouští před tím, než požadavek dosáhne route handleru. Liší se od jiných frameworků tím, že jsou přizpůsobené modulární architektuře NestJS a mohou být definovány na úrovni modulu. Middleware se používají pro manipulaci nebo validaci požadavků, jako je autentizace, logování nebo správa CORS.

1.3 3. Co jsou to namespacy v PHP a jak využivají autoloading? Jak může autoloading zjednodušit práci s velkými kódy?

Namespacy v PHP umožňují organizaci tříd, funkcí a konstant do oddělených prostorů jmen, což předchází konfliktům názvů. Autoloading v PHP načítá třídy automaticky, když jsou potřeba, bez nutnosti manuálního zahrnutí souborů. To výrazně zjednodušuje práci s velkými kódy, protože minimalizuje chyby při importu a zlepšuje čitelnost kódu.

1.4 4. Jaký je rozdíl mezi Guards a Interceptors v NestJs? Jak byste použili Guards a Interceptors k zajištění bezpečnosti a logování ve vaši aplikaci?

Guards a Interceptors jsou schopni přidat dodatečné chování během zpracovávání určitých požadavků. Guard jak už z názvu vyplývá slouží k auth účelům a Interceptor může být použit k logovacím účelům, třeba když chceme vědět kolik sekund daný request trval apod.

1.5 5. Vysvětlete rozdíl mezi == a === v PHP. Kdy byste měli použít každý z těchto operátorů?

Operátor `==` v PHP porovnává hodnoty bez ohledu na jejich typy, což znamená, že může dojít k automatickému přetypování. Operátor `===` porovnává jak hodnoty, tak i jejich typy, takže je přísnější. `==` by měl být použit tam, kde nezáleží na typu proměnných, zatímco `===` je lepší pro případy, kdy je důležité, aby porovnávané proměnné měly nejen stejné hodnoty, ale i stejné typy.

Takže je to v podstatě obdoba toho co nabízí programovací jazyk JavaScript.

1.6 6. Jak funguje systém Dependency Injection v NestJS? Jak může DI zjednodušit testování a správu závilostí v aplikaci?

Dependency Injection (DI) v NestJS umožňuje injektování závislostí (jako jsou služby, repository atd.) do tříd, aniž by bylo nutné je vytvářet ručně. Tedy následuje "Inversion of Control" ze SOLID design patternů, jenž odstranuje vznik tight couplingu a výsledný software je poté více přizpůsoben pro psaní unit testů ve formě mocků apod.

1.7 7. Vysvětlete rozdíl mezi trait a interface v PHP. Kdy byste použili trait a kdy byste použili interface?

Trait v PHP je způsob, jakým můžeme sdílet metody mezi třídami bez nutnosti dědění. Interface definuje smlouvu, kterou musí implementovat každá třída, která rozhraní používá. Trait byste použili, když chcete sdílet konkrétní implementaci metod mezi různými třídami, zatímco interface použijete, když chcete zajistit, aby různé třídy implementovaly určité metody, ale bez sdílení konkrétní implementace.

1.8 8. Co je to DTO a jaký je jeho účetl v NestJS? Jak byste použili DTO k validaci a transformaci dat v příchocích a odchozích HTTP požadavcích?

DTO (Data Transfer Object) je objekt, který se používá k přenosu dat z bodu A do bodu B v backendových službách. Na úrovni DTO lze provádět validaci dat skrze dekorátory. Například JS knihovna "class-validator".

1.9 9. Jak funguje sessions a cookies v PHP? Jak můžete v PHP pracoval s sessions a jak zajistíte bezpečnost při práci s cokkies?

Sessions v PHP umožňují ukládání dat na straně serveru pro daného uživatele, zatímco cookies jsou malé textové soubory uložené na straně klienta. Sessions se obvykle identifikují pomocí session ID, které je uloženo v cookie. Pro práci se sessions v PHP stačí použít funkci `session_start()`, po které můžete ukládat a přistupovat k datům v `_SESSION` superglobálu. Pro bezpečnost při práci s cookies byste měli zajistit, aby byly cookies označeny jako `HttpOnly` a `Secure`, aby nebyly přístupné ze skriptů a byly odesílány pouze přes HTTPS.

1.10 11. Co jsou to dekorátory? K čemu slouží? Lze je použít ve Vanilla JS?

Dekorátory jsou speciální funkce, které mohou být použity k modifikaci chování tříd nebo metod. Dekorátory se používají v hojné míře v Angularu a Nestjs pro úpravu chování na úrovni tříd, aby tato funkčnost byla zachována i v prohlížečích musí dojít k tzv. transkompilaci s přislušnými polyfilly, protože ES5 dekorátory implicitně nepodporuje, jedná se o funkčnost z Next Gen JavaScriptu.

Dekorátory nejsou nativně podporovány ve Vanilla JS, ale lze je použít s nástroji jako je TypeScript nebo Babel, které přidávají podporu pro tuto funkcionalitu.

1.11 12. Co je to Typescript, jaké má výhody a nevýhody oproti Vanilla JS?

TypeScript je nadmnožina JavaScriptu, která přidává statické typování, což umožňuje lepší kontrolu nad zdrojovým kódem během vývoje.

TypeScript je kompilován do nějaké verze JS od (ES7, ES6, ES5) záleží na konfiguraci bundleru Webpack.

ES5 neboli Vanilla JS je dynamický orientovaný jazyk, kdy chyby lze často odhalit až za běhu programu. A pro větší projekty je lepší mít zdrojový kód pod typovou kontrolou. Jediná nevýhoda Typescriptu je náročnější implementace než s klasickým JS. A proto hodně firem všude píšou any, čímž zahazuje celou nutnost TypeScriptu.