**Konfiguráció Javaban**

**(component scan mechanizmus, beanek magadása explicit módon, Spring non invase mód -> példányosítás saját konfigurációval -> @Configuration; tipikusan egy rétegre csak; legjobb gyak: saját beanek componen scanel, 3rd partyk saját configgal**

**@SpringBootApplication annotáció** – egy metaannotáció = rajta is vannak további annok; egyik ilyen a **@ComponentScan** -> az **adott csomag és alcsomagjai összes springes beanjét megtalálja**, vagyis az olyan **amiken van Component, Sevice, Repository** vagy **Controller** (component scan mechanizmus)

**ha mi nem akarunk component scant,** hanem explicit módon meg akarjuk adni a beanjeinket, akkor kell java konfiguráció!

ha a service-einkből nem akarunk Springes dolgokra hivatkozni, nem akarjuk rátenni az @Service-t

a Spring frameworkot ilyenkor **non invasive módon** használjuk, **nem hivatkozunk rá a saját osztályunkból**

ebben az esetben egy @Configuration-el ellátott osztályban megadhatjuk, hogy hogyan példányosítja le a Spring az adott beanünket

**tipikusan rétegenként szoktuk egy osztályban kitenni a @Configuraiton annotációt** (nem az összesben)

**legjobb gyakorlat: saját beanek component scannel, 3rd party libraryk java configgal;** egy komoly alkalmazásba akár több 100 service is van, nem lenne jó mindet magunk példányosítani

**Java konfigurációs példa**

**(no @Service no component scan, de legyen Springes bean -> @Configuration és @Bean egy konstruktorra = explicit példányosítás; mikor jó? -> paraméterezetten példányosíytás; visszaadott példány elhelyezése az AC-ben)**

**nincs rajta a @Service, nem fogja felolvasni a component scan**

hogy **mégis Springes bean legyen**, az osztály legyen **@Configuration** annotációval, legyen benne egy **példányosítító metódus, @Bean-el**

ennek hatására a Spring meghívja a metódust, és **amit ez visszaad, elhelyezi az applicaiton contextben(konténerben)** -> ugyanúgy, mintha lenne rajta @Service; csak ez így **explicit példányosítás**; ha paraméterezetten kell példányosítani és azokat is még be kéne olvasni, akkor jó gyakorlat lehet



**Java konfigurációs példa - gyakorlat**

**HelloServiceből levesszük a @Service-t** -> a HelloServicenek így nincs függősége a Springre

EmployeesApplicaiton belül saját példányosítás és **rá @Bean**

**nem kell külön @Cofiguraiton,** mert az része az osztályon rajta lévő @SpringBootApplicationnek

legjobb gyakorlat szerint érdemes a saját komponenseinket a componentscan által felolvasni (ha rajta van a @Controller, @Service stb.), és a 3rd party libraryket java configgel ellátni (@Bean) -> ezért vissza is raktuk a kikommentelt @Service-t

**ja nincs se config, se @Service, hibaüzit fogunk kapni** -> nem tudja létrehozni a HelloController Beant, mert van egy hivatkozása a HelloServicere de az @Service annotáció és Bean hiányában a Spring számára nem ismert

**Kérdések**

*Minek a része a @ComponentScan és mire jó?*

*Mit jelent az, hogy non invasive módon használjuk a Springet pl. a Service osztályban?*

*Hogy érjük el, hogy mégis springes bean legyen a Serviceből?*

*Mikor jó gyakorlat ez az explicit példányosíás?*

*Mi történik, ha egy controllernek van olyan service kapcsolata, amire nincs se saját config, se componentscan?*