Mik a függőséginjektálás főbb előnyei, és hogyan támogatja a Spring? Milyen élettartama lehet egy springes beannek? Hogyan könnyíti meg a JPA használatát a Spring Data JPA? Hogyan támogatja a tranzakciókezelést a Spring?

DIA 7 10 oldal

Az objektumok nem drótozzák be az általuk használt konkrét osztályokat, hanem tagváltozó, metódus- vagy konstruktorparaméter formájában egy ún. injektortól kapja meg. Lényegében csak leírjuk, hogy milyen fajta az a dolog amit kapni fog amit persze a későbbiekben majd több helyen több modón implementálhatunk.

A Spring támogat tagváltozó, konstruktor és setterinjektálást. Ezeket a spring által kezelt osztályokat beanneknek nevezzük. Beanekbe injektált bean-eket már kollaborátornak nevezzük.

DIA 7 18 oldal

Alapból minden bean egy singleton az app contexten belül. Ezen kívül lehet:

1. **prototype (minden injektálási ponton új példány)**
2. **request (egy HTTP kérés idejéig) > session (egy HTTP session idejéig)**
3. **application (a webalkalmazáshoz kötött ◊több app. contextre közös)**
4. **websocket (egy websocket élettartamához kötött**

DIA 7 40 oldal pár mondat

Segítségével az adatelérési kód nagyrésze megspórolható. De írhatunk saját entitás specifikus repositorykat. Lényegében könnyítik a JPA használatát és lehetővé tesz egy gyorsabb és hatékonyabb fejlesztést. Ezt automatikus repository-k generálásával, név alapú lekérdezésekkel , a változások lekövethetőségét teszit lehetővé az auditinggal.

DIA 7 47 oldal

Egységes API van a tranzakciókezelésre, ami mögött persze bekonfigolhatjuk a tranzakció menedzsert. A tranzakciókezelés müködhet JDBC connection szinten, használhatja a JPA EntityTransaction-jét, elérhet elosztott tranzakciómenedzsert JTA API-n keresztül és használhatja a JDO API-t ami a JPA legrégebbi alternatívája. A tranzakciókezelés lehet programozott vagy dekleratív. A programozottat a Spring által adott API-n keresztül a kodból indítjuk és zárjuk . A dekleratívot metodus szinten annotációkkal vagy xml-ben indítjuk vagy zárjuk, illetve itt metodusnál kisebb egységekről nem redelkezhetünk. Illetve A Spring számos annotációt biztosít a tranzakciókezelés egyszerűbb konfigurálásához, például @Transactional annotációval megjelölt metódusokhoz, amelyek tranzakcióban futnak.

A @Transacional-nak a következők lehetnek a praméterei:

1. **rollbackFor: milyen kivételek esetén legyen rollback**
2. **timeout**
3. **isolation: izolációs szint,**
4. **value: tranzakciómanager azonosító**

5. **propagation: mi történjen, ha tranzakcionális**
6. **metódusból másik tranzakcionális metódust hívunk**

A Spring támogatja különböző tranzakciókezelési stratégiákat, például a tranzakcióakik módosítási, olvasási és olvasási/szöveges tranzakciók, valamint a tranzakciók propagálásának és izoláltsági szintjének konfigurálását.