spring boot

bean:

אובייקט במערכת. מגדירים את ה bean בתוך קובץ config.

bean scope: ביצירת bean ניתן להגדיר לו מספר scopes בהם הוא יחיה:

1. singleton: ה bean יווצר פר application context, כלומר הוא יהיה יחיד בכל application context, כלומר יכול להיות מצב שהוא יאותחל מספר פעמים באותו פרוייקט.
2. prototype: ה bean יווצר כל פעם מחדש
3. RequestScope:
4. SessionScope:
5. ApplicaionScope:
6. WebSocketScope:

Dependency Injection:

1. Field by name injection:

- הזרקת אובייקט לפי שם הפוקנציה שמגדירה את ה bean. לדוגמא:

```
@Resource(name="namedFile")
private File defaultFile;
```

הפוקנציה מגדירה את ה bean:

```
@Bean(name="namedFile")
public File namedFile() {
    File namedFile = new File("namedFile.txt");
    return namedFile;
}
```

2. Field by type injection:

- הזרקת אובייקט לפי הסוג של ה bean. במקרה זה באנוטציה @Resource לא מכניסים ערך.

3. Field by qualifier injection:

- כאשר מוגדר יותר ממימוש אחד ל bean, ניתן להזריק את הbean המתאים על ידי שימוש באנוטציה @Qualifer כפי שמתואר בדוגמא למטה. במצב בו אין הגדרה ברורה למימוש הנכון יזרק *NoUniqueBeanDefinitionException*.

```
@Resource
@Qualifier("defaultFile")
private File dependency1;

@Resource
@Qualifier("namedFile")
private File dependency2;
```

:@Autowired vs @Inject vs @Resource

1. כאשר בעיצוב הפרוייקט מסתמך בעיקר על שימושים ב interfaces ו- abstract class, אז משתמשים ב @Autowired או @Inject, כך כאשר יהיה שינוי בפרוייקט, ניתן להחליף את המחלקה באופן שגורר שינוי מינימלי. בתרחיש זה סוג ההזרקה הדיפולטי יהיה Field by type injection
2. כאשר בפרוייקט יש מודולים שונים עם התנהגיות שונות עם לוגיקות מופרדות, אז עדיף להתשמש ב @Resourse, כאשר סוג ההזרקה הדיפולטי יהיה Field by name injection
3. אם נרצה הלהשתמש בהזרקה על ידי סביבת spring צריך להשתמש ב @Autowired.

Scenario	@Resource	@Inject	@Autowired
Application-wide use of singletons through polymorphism	✗	✔	✔
Fine-grained application behavior configuration through polymorphism	✔	✗	✗
Dependency injection should be handled solely by the Jakarta EE platform	✔	✔	✗
Dependency injection should be handled solely by the Spring Framework	✗	✗	✔

:Transaction management

ניתן לבצע מספר פעולות מול ה db תוך כדי הבטחה שהן כולן תחת אותה טרנזקסיה, על ידי שימוש באנוטציה של @Transaction מעל הפונקציה.
במצב כזה, מאחורי הקלעים, נלקח חיבור ל db מה connection pool, והטרנזקציה

מתבצעת תחתיו.
בעיות אפשריות:
- כאשר נירצה להשתמש ב Transaction@ בפונקציה שמשלבת פניות למספר מקורות i/o כמו db ו api חיצוני, הפונקציה תחזיק חיבור ל db כל עוד הקריאה ל api לא תיגמר, ובכך יכולים להיגמר הconnections ל db.

ניתן להשתמש באופן עצמאי (לא על ידי הדיפולט של spring) על ידי שימוש בממשק של TransactionTemplate.
הממשק מכיל פונקציה ()execute אשר הקוד שבתוכה רץ בתוך טרנזקציה אחת, ואם משהו נכשל מתבצע rollback .