הכנת סביבת העבודה

כדי למנוע תקלות של הרשאות כתיבה וכדומה C יצירת תיקייה בשם שלכם בכונן
c:/eldar

.מותקן כבר על המחשבים של הכיתה – JDK התקנת

.אבל מה שנוח לכם. להוריד לתיקייה שיצרתם (ללא התקנה) Eclipse אני עובד עם – IDE הורדת

workspace יצירת תיקיית

c:/eldar/workspace

מי שעובדים עם Eclipse להוריד ZIP ולא קובץ התקנה:
Eclipse IDE for Java Developers

הכנת סביבת העבודה

כדי למנוע תקלות של הרשאות כתיבה וכדומה C יצירת תיקייה בשם שלכם בכונן
c:/eldar

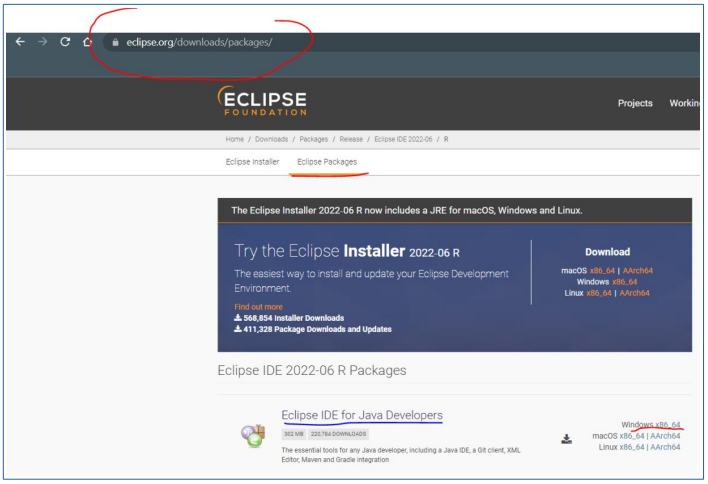
.מותקן כבר על המחשבים של הכיתה – JDK התקנת

.אבל מה שנוח לכם. להוריד לתיקייה שיצרתם (ללא התקנה) Eclipse אני עובד עם – IDE הורדת

workspace יצירת תיקיית

c:/eldar/workspace

מי שעובדים עם Eclipse להוריד ZIP ולא קובץ התקנה:
Eclipse IDE for Java Developers



אפשרויות עם בנאים:

אפשר להגדיר כמה בנאים שרוצים @Component בכל מחלקה עם

- בנאי אחד עם פרמטרים: מקבל autowired אוטומטית (spring תשתמש בו ותזריק)
- ספרינג תשתמש בבנאי - autowired בנאי אחד ללא פרמטרים ועוד בנאים עם פרמטרים אבל ללא שלא מקבל פרטמרים.
- ספרינג תשתמש בבנאי שמקבל - autowired בנאי אחד ללא פרמטרים ובנאי עם פרמטרים ועם פרמטרים ותזריק. (על השדות autowired מותר בנוסף לתת)
- ספרינג תשתמש בבנאים עם - autowired מספר בנאים שמקבלים פרמטרים אבל רק אחד עם autowired ותזריק.
- ספרינג תזרוק שגיאה שאין בנאי ברירת - autowired מספר בנאים שמקבלים פרמטרים אבל ללא מחדל.
- אסור יותר מבנאי אחד עם autowired.

אסטרטגיה לביצוע Autowired

- אז מוזרק (type לפי) אם יש התאמה אחת בלבד
- אם יש Primary אם יש יותר מאחת אז לפי
- אם יש כמה אפשרויות ואין Primary אז לפי Qualifier
- לפי שם השדה

שתי דרכים להגדיר bean

1. להגדיר class עם אנוטציה Component – כשמבקשים bean ה container מפעיל בנאי
2. להגדיר מתודה עם אנוטציה Bean – כשמבקשים bean ה container מפעיל את המתודה

Annotations

@ComponentScan

שמים על class למטרת קונפיגורציה, כדי להגדיר סריקת מחלקות כדי לאתר הגדרות של beans

@Configuration

שמים על class למטרת קונפיגורציה, כדי להגדיר שה class יכול להכיל bean methods

@PropertySource

שמים על class למטרת קונפיגורציה, כדי להגדיר מיקום לקובץ properties

@Component

שמים על מחלקה כדי להגדיר bean מהסוג של המחלקה

@Bean

שמים על מתודה כדי להגדיר bean מהסוג שהמתודה מחזירה (בתוך class שמכיל אנוטציית Configuration)

@Scope("prototype")
@Scope("singleton")

קונפיגורציה להגדרת bean – האם יהיה סינגלטון או פרוטוטייפ

@Autowired

הוראה להזריק bean שנדרש ל bean הנוכחי:

1. CTOR
2. field
3. setter

@Qualifer

מצורף בהקשר של Autowired כשיש יותר מהתאמה אחת ורוצים לציין איזה להזריק

@Primary

מצרפים להגדרה של bean כדי לקבוע שהוא ההתאמה העיקרית במקרה שיש כמה אפשרויות.

@Value

מצרפים לשדה או פרמטר בבנאי או מתודה כדי להזריק ערך מתוך קובץ properties

@Lazy

שמים על הגדרה של bean שהוגדר כ singleton על מנת לשנת את ההתנהגות ברירת מחדל של טעינה Eager. כלומר ה bean לא ייווצר עד שמישהו מבקש אותו. לא רלוונטי ל prototype (תמיד lazy)

Spring AOP – Aspect Oriented Programming

AOP – פרדיגמה בתכנות שמתמקדת ב aspects (היבטים) – שהם נושאים לטיפול שמשתרעים על יותר ממודול אחד של האפליקציה.

לכל מודול באפליקציה יש עניין ליבתי core concern – למשל פעולות מול database

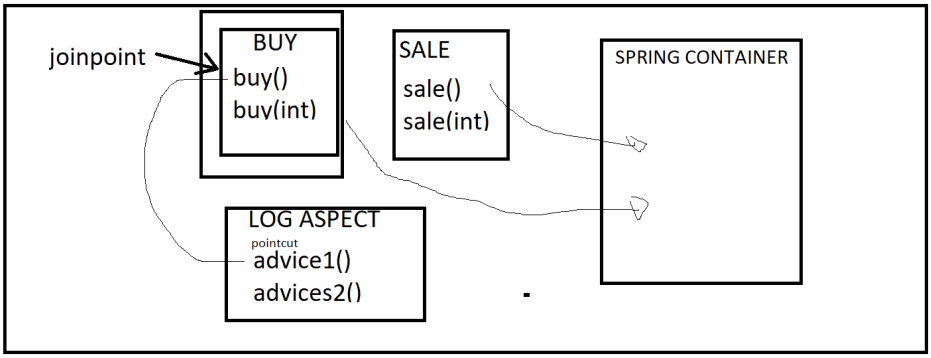
aspect הוא עניין שיש לטפל בו אבל הוא חוצה את האפליקציה – cross cutting concern

למשל:

Aspect are modules for cross cutting concerns of the application
mainly for:

- logging
- security
- statistic information

<u>Terminology</u>
aspect – a class with advice methods
advice – a method in aspect class to be run with joinpoints
joinpoint – a businesses method intercepted by advice
pointcut – a text expression for matching advice to jpoinpoints



Advice types:

1. Before
2. After
3. After Returning
4. after Throwing
5. Around

<u>Annotations</u>
@Aspect – on aspect class
@Before – on advice method
@EnableAspectJAutoProxy – on configuration class

pointcut expression

execution(
modifers
return type
declaring type pattern
method name pattern
parameters pattern
throws pattern
)

// 1. modifiers - any if not specified
// 2. return type - * = any [mandatory]
// 3. declaring type (package.class.) - any if not specified
// 4. method pattern [mandatory]
// 5. parameter pattern [mandatory]
// 6. throw pattern

parameters
() - no parameters
(type1, type1) – specific parameters
(*) – one parameter of any type
(..) – 0 or more parameters of any type

How to Define Pointcuts

```
// POINTCUTS

// no annotations in business
@Pointcut("execution(java.lang.String divide(Integer, Integer))")
public void div() {
}

// for annotations on methods
@Pointcut("@annotation(app.core.aspects.annotations.MyLogAnnotation)")
public void annotatedMethod() {
}

// for annotations on class
@Pointcut("@target(app.core.aspects.annotations.MyLogAnnotation)")
public void annotatedClass() {
}
```