**Spring Boot**

Spring Boot – ը Spring Framework – ի կոմպոնենտ է։

Spring framework – ը ունի մի շարք անհարմարություններ՝ config – ների համար մեծ քանակույամբ կոդ, dependecy – ները և դրանց համատեղելիությունը շատ հաճախ դրանց մեջ լինում են կոնֆլիկտներ, app – ի deployment – ը (այդ թվում tomcat – ի ձեռքով configuration - ը)։ Spring boot – ը լուծում է նշված և այլ խնդիրներ։ Կարող է ավտամատ կատարել coniguration – ը օգտագործվող տարբեր գործիքների համար օր՝ (Hibernate, Security, Timeleaf և այլն)։ App – ի war ֆայլում հավաքվելը և deploy – ը հեշտանում են այնքանով որ Spring boot – ը իր մեջ պարունակում է web սերվեր ընդ որում կարելի է ընտրել սերվերը tomcat կամ այլ։ Այսինքն app – ն run է լինում main մեթոդից սովորական java application – ի նման, իսկ մնացած գործողությունները տեղի են ունենում տակից։ Սերվերի վրա deploy եղած app – ի jar – ը հավաքելը ձեռքով կատարվում է cmd – ից maven – ի միջոցով: Դրա համար պրոյեկտի ամենավերևի կատալոգում՝

(jar ֆայլում պրոյեկտը հավաքելու համար) – mvnw package

(հավաքված jar – ը իր մեջ պարունակում է ներդրված սերվեր և գտնվում է target folder – ի մեջ, jar - ի ստարտի համար cmd – ով նշված folder-ում -

Java -jar JarFileName.jar)

Dependency – ներ դեպքում Spring Boot – ը տրամադրում է այսպես կոչված starter – ներ, որոնք իրենցից ներկայացնում են dependency – ներ խմբավորված մեկ մեկ dependency – ի մեջ։ Նման մոտեցումը թույլ է տալիս խուսափել շատ dependency – ների միջև անհամատեղելիություններից։ Օր՝ spring-boot-starter-web որը իր մեջ պարունակում է sping core, web, web-mvc, json – ի հետ աշխատող կախվածություններ, tomcat և այլն։

Բացասական կողմը այն է որ տակից տեղի ունեցող գործողություններ են շատ և որպես հետևանք error – ների ծագումը/պատճառների բացահայտումը կարող է բարդանալ։

App – ն ստեղծելու համար օգտագործվելու է Spring Initializr – ը, որը իրենից ներկայացվնում է web application, որի միջոցով ստեղծվում է app – ի կարկասային մասը։

App – ն ստարտ անող կլասսում @SpringBootApplication անոտացիան ցույց է տալիս որ app – ն Sprin Boot է։ Բացի դրանից անոտացիան իր մեջ ներառում է հտևյալ անոտացիաները՝

* @SpringBootConfiguration որը սահմանում է կլասսը որպես կոնֆիգուրացիոն, @Configuration – ի հատուկ տեսակ է
* @EnableAutoConfiguration սահմանում է ավտոկոնֆիգուրացիան, որը հաղորդում է Spring Boot – ին app – ում օգտագործված կոմպոնենտների ավտոմատ կոնֆիգուրացիյի անհրաժեշտության մասին
* @ComponentScan միացնում է կոմպոնենտների սքանը։ Մեխանիզմ որը թույլ է տալիս հայտնաբերել կոմպոնենտներ ստեղծող անոտացիաները՝ @Component, @service և այլն, բայց այն պայմանով որ ստարտ անող կլասսը պետք է գտնվի root մակարդակում այսինքն մնացած package – ները և կլասսները պետք է լինեն դրա հետ հույն կամ ավելի ցածր մակարդակում:

Կլասսը ունի նաև main() մեթոդ որի պարունակությունը հիմնականում նույնն է։ կանչվում է SpringApplication.run(Class clazz, String args):

Boot – ում application.properties ֆայլը default տեսանելի է և կարիք չկա դրա path – ը առանձին նշել @PropertySource անոտացիայով:

Եթե անհրաժեշտ են template շաբլոններ, կամ նկար և այլ ռեսուրսներ, դրանք համապատասխանաբար պետք է լինեն templates, static folder – ներում resources – ի տակ: Boot app – ն հավաքվում է jar ֆայլում, նախկինում նման ֆայլերը պահվում էին web-app folder - ում քանի որ app – ները հավաքվում էին war ֆայլում, միայն war – ում է որ հաշվի է առնվում web-app folder – նը: