**Spring boot devtools:**

kaynak kodda yaptığımız değişikliklerin otomatik olarak uygulanmasını sağlar, böylece uygulamamızı manuel olarak yeniden başlatmamıza gerek kalmaz. intellij comminiti edition'da bu ayarı yapmak için ctrl+alt+s ile preference kısmından "build project automatically" özelliğini ardından "advanced settings">"allow aouto-make to start even if..." özelliğini aktif etmemiz gerekir son olarak pom.xml dosyasına spring-boot-devtools dependecy'İni eklememiz gerekir.

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
</dependency>

**Spring boot Actuator:**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>

Uygulamayı izleme ve yönetme, uygulamanın başvuru durumunu izleme ve bazı uygulama metriklerine erişmeyi sağlar. Uygulamayı izlemek ve yönetmek için uç noktalar(endpoints) oluşturur. DevOps işlevselliğini kolayca elde etmeyi sağlar. Projeye dahil etmek için yalnızca pom.xml dosyasına dependecy eklemek yeterli olur.

**Otomatik eklenen endpointler:**

Tüm endpointleri açığa çıkartmak için src>main>resources/application.properties

Dosyasına management.endpoints.web.exposure.include=\* komutunu eklemek yeterli olur, böylece tüm endpointler tek seferde açığa çıkar.

/health: uygulamanın sağlıklı çalışıp çalışmadığı konusunda bilgi sağlar.

/auditevents: denetim olaylarının listesini verir.

/beans: projede tanımlı spring boot itemlerinin (örneğin sınıflar) bir listesini verir.

/mappings: maplemelerin listesini verir.(örn:@RequestMapping)

/info: default olarak boş gelen bir endpointtir. Bilgi eklemek için src>main>resources/application.properties dosyasına  
info.app.name=My app

İnfo.app.decscription=A demo app

İnfo.app.version=1.0.0  
şeklinde bilgi eklenebilir.

**Actuator’da güvenlik:**

Application.properties dosyasına   
spring.security.user.name=scott  
spring.security.user.password=tiger

Bilgileri eklenerek endpointlere erişim için kullanıcı adı ve şifre zorunlu hale getirilir.

**Application Properties**

Proje dosyasında src main altında bulunan “resources” klasörü altında bulunan application.properties dosyasına eklemek istediğimiz bilgileri yazabiliriz.  
ÖRNEK:

Coach.name=Micky Mouse

Team.name=The Mouse Club

Eklediğimiz bu bilgilere erişmek için projemize RestController ile enjekte etmemiz gereklidir.

ÖRNEK:

@Value("${coach.name}")  
private String coachName;  
@Value("${team.name}")  
private String teamName;

Bu kodlar ile properties dosyasına eklediğimiz bilgileri değişkenlere atayabiliriz, burada önemli olan “@Value” annotation’u dur. Daha sonra değişkenlere atadığımız bu değerleri görebilmek için endpoint tanımlamamız gerekir.

@GetMapping("/teaminfo")  
public String getTeamInfo(){  
 return "Coach: "+coachName+", Team Name: "+teamName;  
}

**Spring boot server configuration**

Log ayarlarını src/main/resources klasörü altında bulunan “application.properties” dosyası içinden yapabiliriz.

**Logging Levels:**

**TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF**

**ÖRNEK:**

**# Log levels severity mapping**

Logging.level.org.springfreamwork=DEBUG

Logging.level.org.hibernate=TRACE

Logging.level.com.luv2code=INFO

**# Log file name**

Logging.file.name=my crazy-stuff.log

Logging.file.path=c:/myapps/demo

**# http server port**

Server.port=7070

**#Context path of the application**

Server.servlet.context-path=/my-silly-app

**# Default http session time out**

Server.servlet.session.timeout=15m

**# JDBC URL of database**

Spring.datasource.url=jdbc:mysql://localhost:3306/ecomerce

**#Login username of the database**

Spring.datasource.username=scott

**#Login password of the database**

Spring.datasource.password=tiger