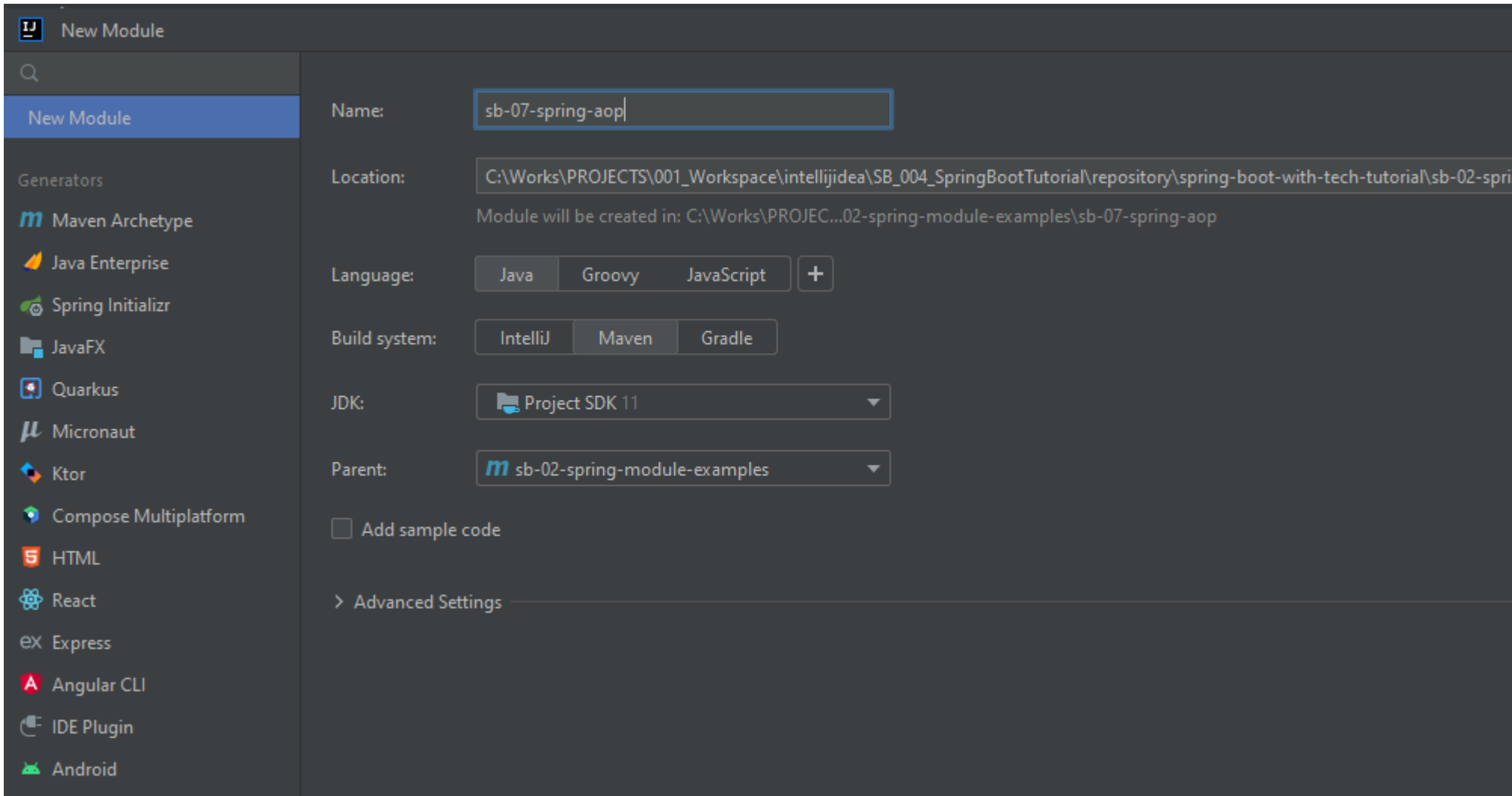
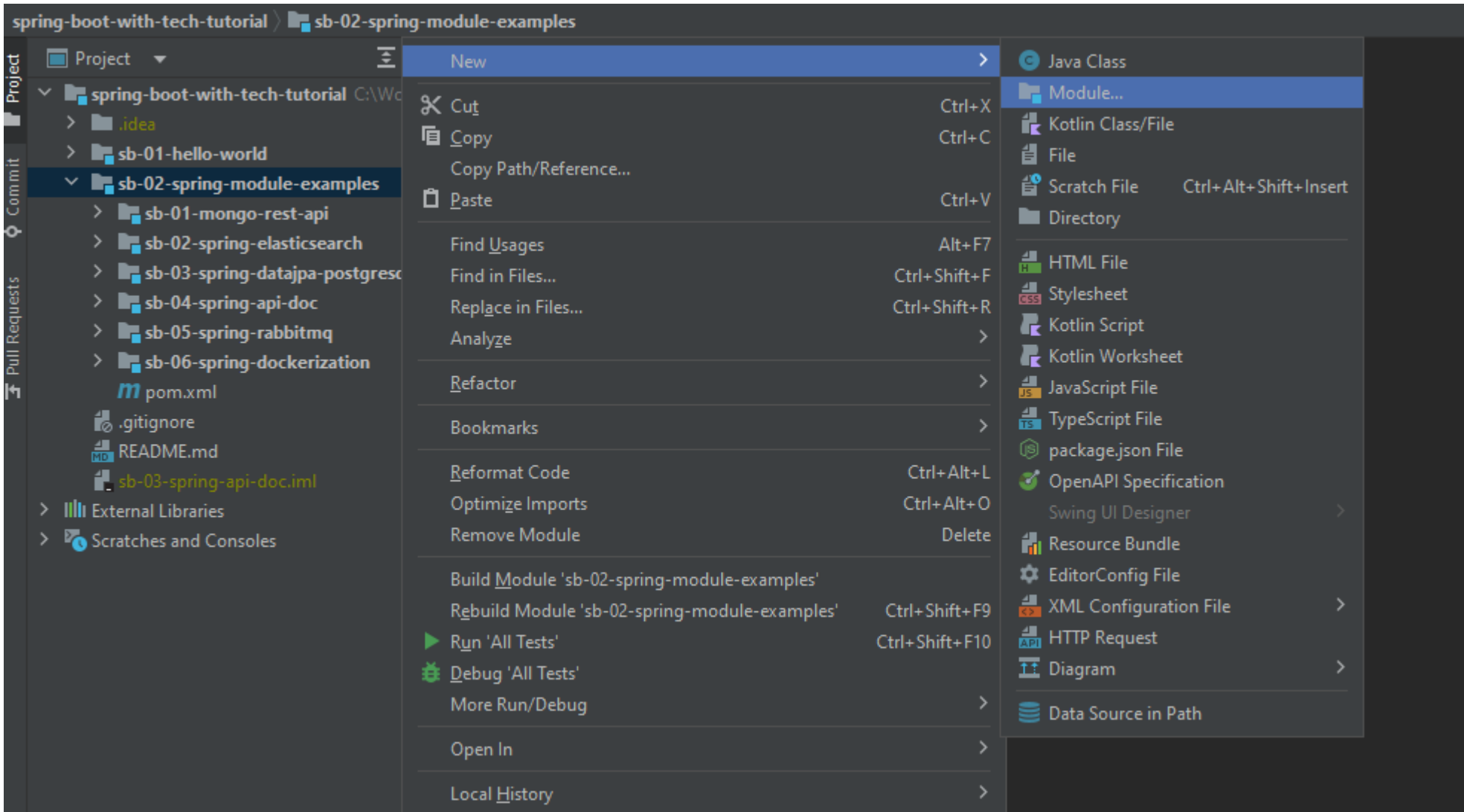
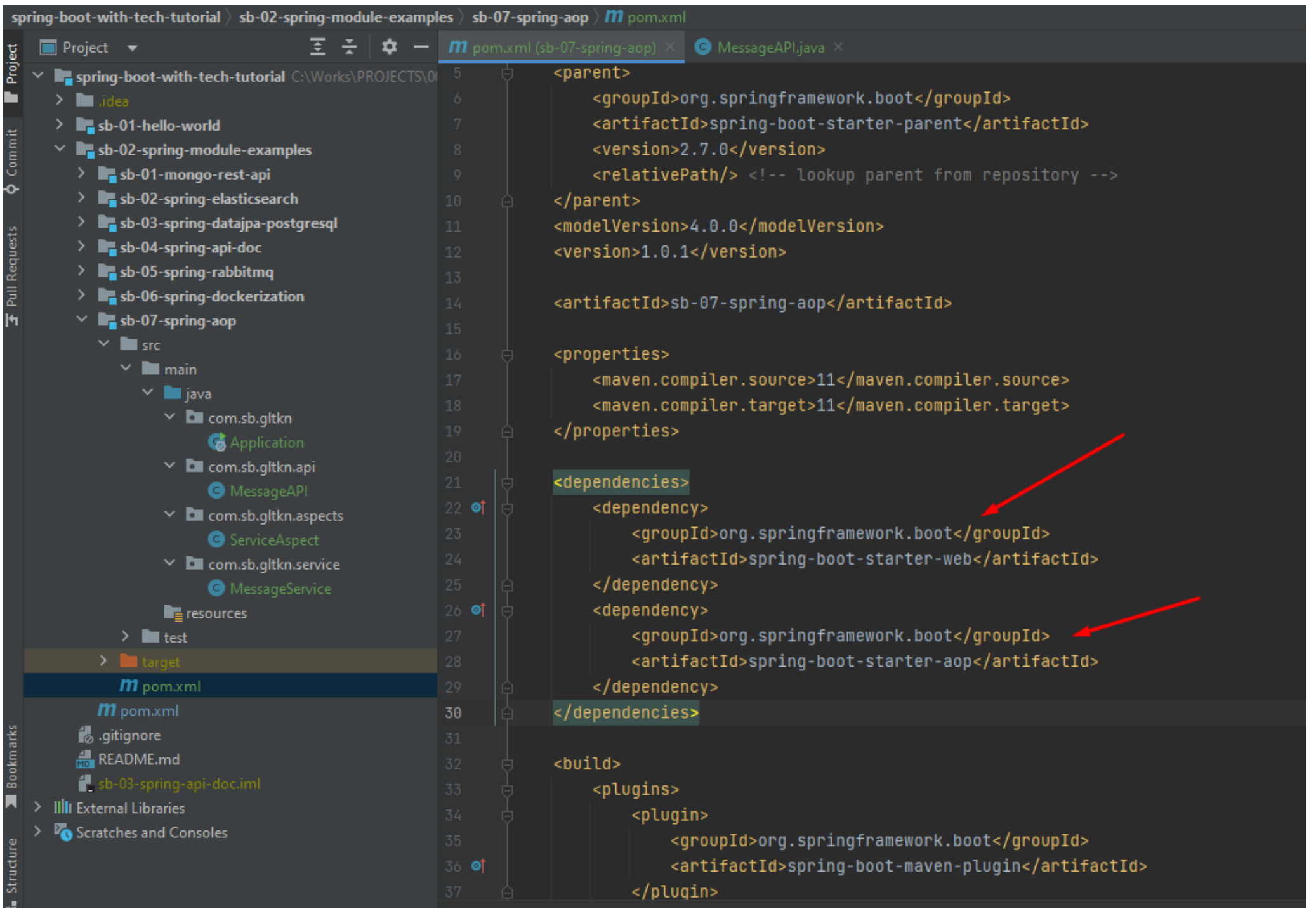


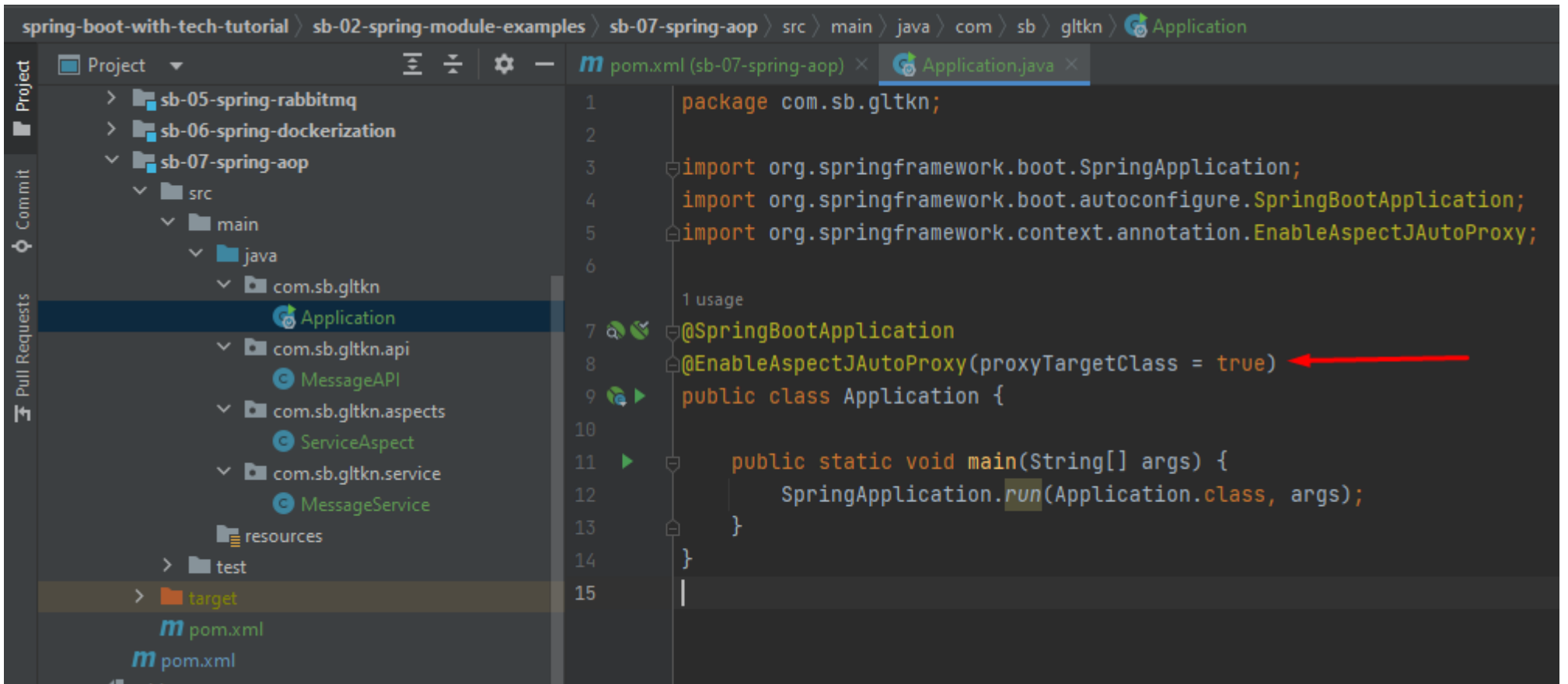
Bu bölümde SpringBoot ile Aspect Oriented Programming(AOP) örneği uygulayacağız.





```
5 <parent>
6   <groupId>org.springframework.boot</groupId>
7   <artifactId>spring-boot-starter-parent</artifactId>
8   <version>2.7.0</version>
9   <relativePath/> <!-- lookup parent from repository -->
10 </parent>
11 <modelVersion>4.0.0</modelVersion>
12 <version>1.0.1</version>
13
14 <artifactId>sb-07-spring-aop</artifactId>
15
16 <properties>
17   <maven.compiler.source>11</maven.compiler.source>
18   <maven.compiler.target>11</maven.compiler.target>
19 </properties>
20
21 <dependencies>
22   <dependency>
23     <groupId>org.springframework.boot</groupId>
24     <artifactId>spring-boot-starter-web</artifactId>
25   </dependency>
26   <dependency>
27     <groupId>org.springframework.boot</groupId>
28     <artifactId>spring-boot-starter-aop</artifactId>
29   </dependency>
30 </dependencies>
31
32 <build>
33   <plugins>
34     <plugin>
35       <groupId>org.springframework.boot</groupId>
36       <artifactId>spring-boot-maven-plugin</artifactId>
37     </plugin>
38   </plugins>
39 </build>
```

Bu örnekte **service** paketi altında herhangi bir class'tan çağrılan **createMessage** metodunun öncesinde ve sonrasında araya girip işlemler yapacağız.



```
1 package com.sb.gltn;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.EnableAspectJAutoProxy;
6
7 @SpringBootApplication
8 @EnableAspectJAutoProxy(proxyTargetClass = true)
9 public class Application {
10
11   public static void main(String[] args) {
12     SpringApplication.run(Application.class, args);
13   }
14 }
15
```

```
spring-boot-with-tech-tutorial > sb-02-spring-module-examples > sb-07-spring-aop > src > main > java > com > sb > gltkn > api > MessageAPI

Project
  ▾ spring-boot-with-tech-tutorial C:\Works\PROJECTS\01
    > .idea
    > sb-01-hello-world
    ▾ sb-02-spring-module-examples
      > sb-01-mongo-rest-api
      > sb-02-spring-elasticsearch
      > sb-03-spring-datajpa-postgresql
      > sb-04-spring-api-doc
      > sb-05-spring-rabbitmq
      > sb-06-spring-dockerization
      ▾ sb-07-spring-aop
        ▾ src
          ▾ main
            ▾ java
              ▾ com.sb.gltkn
                Application
              ▾ com.sb.gltkn.api
                MessageAPI
              ▾ com.sb.gltkn.aspects
                ServiceAspect
              ▾ com.sb.gltkn.service
                MessageService
            resources
          > test
        target
        pom.xml

pom.xml (sb-07-spring-aop) × MessageAPI.java ×
1 package com.sb.gltkn.api;
2
3 import com.sb.gltkn.service.MessageService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PostMapping;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10
11 @RestController
12 @RequestMapping("/message")
13 public class MessageAPI {
14
15     1 usage
16     @Autowired
17     private MessageService messageService;
18
19     @PostMapping
20     public ResponseEntity<String> createMessage(@RequestBody String message){
21         return ResponseEntity.ok(messageService.createMessage(message));
22     }
23 }
```

```
spring-boot-with-tech-tutorial > sb-02-spring-module-examples > sb-07-spring-aop > src > main > java > com > sb > gltkn > service > MessageService

Project
  ▾ spring-boot-with-tech-tutorial C:\Works\PROJECTS\01
    > .idea
    > sb-01-hello-world
    ▾ sb-02-spring-module-examples
      > sb-01-mongo-rest-api
      > sb-02-spring-elasticsearch
      > sb-03-spring-datajpa-postgresql
      > sb-04-spring-api-doc
      > sb-05-spring-rabbitmq
      > sb-06-spring-dockerization
      ▾ sb-07-spring-aop
        ▾ src
          ▾ main
            ▾ java
              ▾ com.sb.gltkn
                Application
              ▾ com.sb.gltkn.api
                MessageAPI
              ▾ com.sb.gltkn.aspects
                ServiceAspect
              ▾ com.sb.gltkn.service
                MessageService
            resources
          > test
        target
        pom.xml

pom.xml (sb-07-spring-aop) × MessageAPI.java × MessageService.java ×
1 package com.sb.gltkn.service;
2
3 import org.springframework.stereotype.Service;
4
5     2 usages
6     @Service
7     public class MessageService {
8
9         1 usage
10        public String createMessage(String message) {
11            System.out.println("\ncreateMessage metodu calisti!!!");
12            System.out.println(message);
13            return message;
14        }
15 }
```

```
Project
spring-boot-with-tech-tutorial C:\Works\PROJECTS\0
> .idea
> sb-01-hello-world
> sb-02-spring-module-examples
> sb-01-mongo-rest-api
> sb-02-spring-elasticsearch
> sb-03-spring-datajpa-postgresql
> sb-04-spring-api-doc
> sb-05-spring-rabbitmq
> sb-06-spring-dockerization
> sb-07-spring-aop
  src
    main
      java
        com.sb.gltkn
          Application
          com.sb.gltkn.api
            MessageAPI
            com.sb.gltkn.aspects
              ServiceAspect
            com.sb.gltkn.service
              MessageService
          resources
        test
      target
    pom.xml
    pom.xml
    .gitignore
    README.md

pom.xml (sb-07-spring-aop) x MessageAPI.java x MessageService.java x ServiceAspect.java x
1 package com.sb.gltkn.aspects;
2
3 import org.aspectj.lang.JoinPoint;
4 import org.aspectj.lang.annotation.After;
5 import org.aspectj.lang.annotation.Aspect;
6 import org.aspectj.lang.annotation.Before;
7 import org.springframework.stereotype.Component;
8
9 @Aspect
10 @Component
11 public class ServiceAspect {
12
13     //Bu paket altında bulunan tüm class'lardaki createMessage metodundan hemen önce çalışır.
14     @Before("execution(* com.sb.gltkn.service.*.createMessage(..)")
15     public void beforeCreateMessage(JoinPoint joinPoint){
16         System.out.println("\n[createMessage metodundan önce]");
17         System.out.println("Yakalanan ilk parametre: " + joinPoint.getArgs()[0]);
18         System.out.println("Yakalanan signature: " + joinPoint.getSignature());
19     }
20
21     //Bu paket altında bulunan tüm class'lardaki createMessage metodundan hemen sonra çalışır.
22     @After("execution(* com.sb.gltkn.service.*.createMessage(..)")
23     public void afterCreateMessage(JoinPoint joinPoint){
24         System.out.println("\n[createMessage metodundan sonra]");
25         System.out.println("Yakalanan ilk parametre: " + joinPoint.getArgs()[0]);
26         System.out.println("Yakalanan signature: " + joinPoint.getSignature());
27     }
28 }
```

<http://localhost:8080/message>

POST http://localhost:8080/message

No Enviro

▶ http://localhost:8080/message Exar

POST http://localhost:8080/message

Params Authorization Headers (9) Body Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON

1 {

2 "message": "PostMessage#1"

3 }

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 182

Pretty Raw Preview Visualize

Text

1 {

2 "message": "PostMessage#1"

3 }

```
[createMessage metodundan önce]
Yakalanan ilk parametre: {
  "message": "PostMessage#1"
}
Yakalanan signature: String com.sb.gltkn.service.MessageService.createMessage(String)

createMessage metodu çalıştı!!!
{
  "message": "PostMessage#1"
}

[createMessage metodundan sonra]
Yakalanan ilk parametre: {
  "message": "PostMessage#1"
}
Yakalanan signature: String com.sb.gltkn.service.MessageService.createMessage(String)
```

