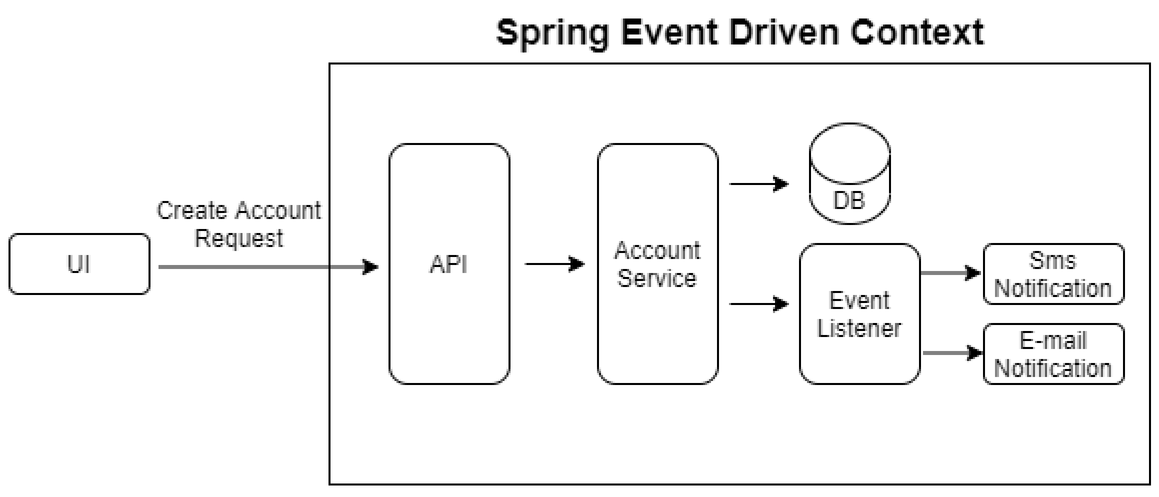
Spring Event Management'tan bahsedeceğiz.

Spring tarafından yönetilen bean'lar iş akış sürecinde meydana gelen bir takım event'leri **ApplicationContext** vasıtası ile *publish* ederler.

Bu event'ler **ApplicationEvent** sınıfından extend etmelidir. Yine Spring tarafından yönetilen **ApplicationListener** interface'ine sahip bean'lar, Spring container tarafından özel olarak tespit edilerek, meydana gelen bu event'lerden haberdar edilirler. Bu sayede ApplicationListener nesneleri, ilgilendikleri event'ler üzerinden iş mantığı ile ilgili görevlerini yerine getirme fırsatı bulurlar.

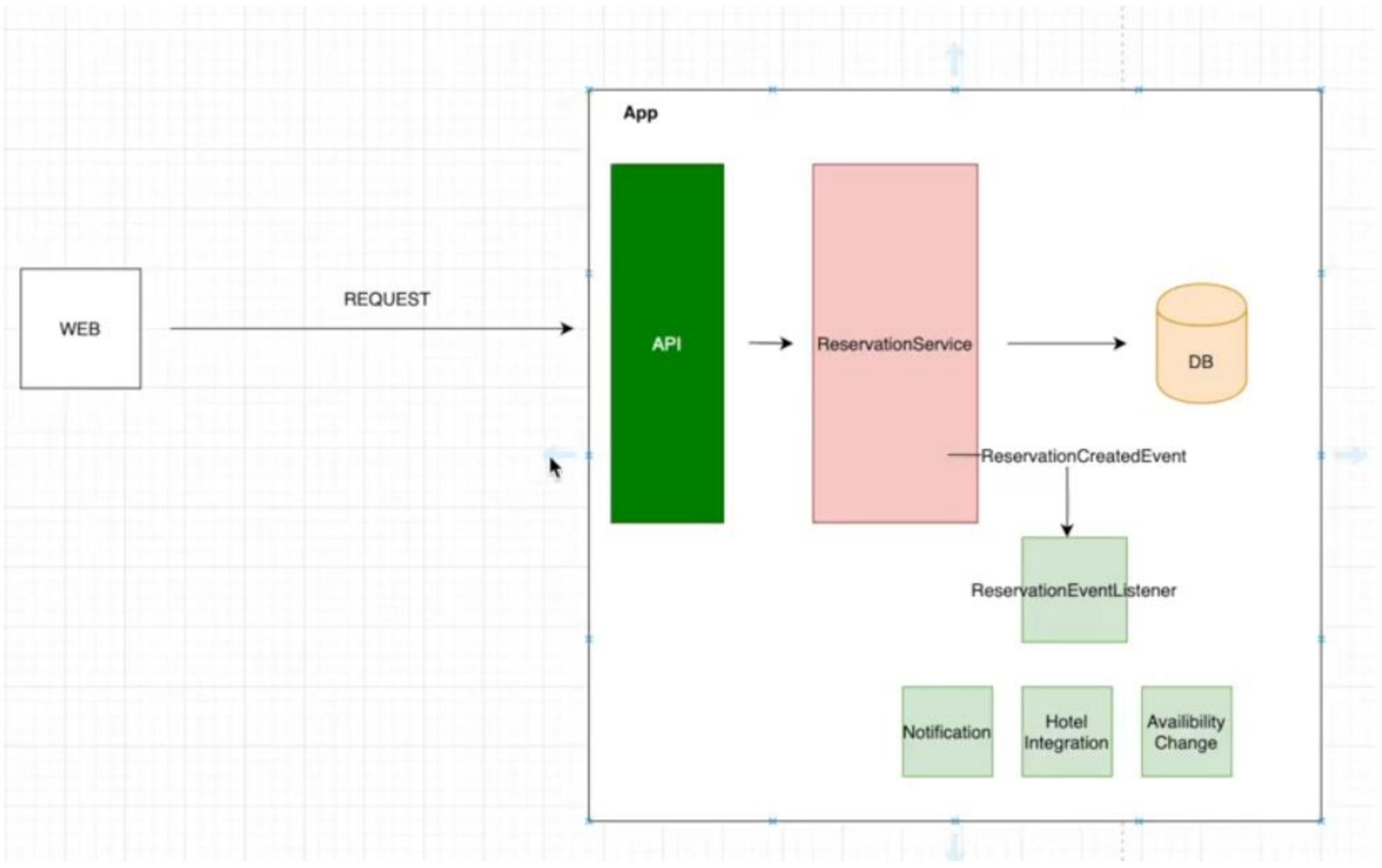
Event Driven uygulamalar bir istek sonrası meydana gelmesi istenen eylemledir.
Örneğin yeni bir hesap oluşturma sonrası kullanıcıya e-mail veya sms ile bildirim yapılması.



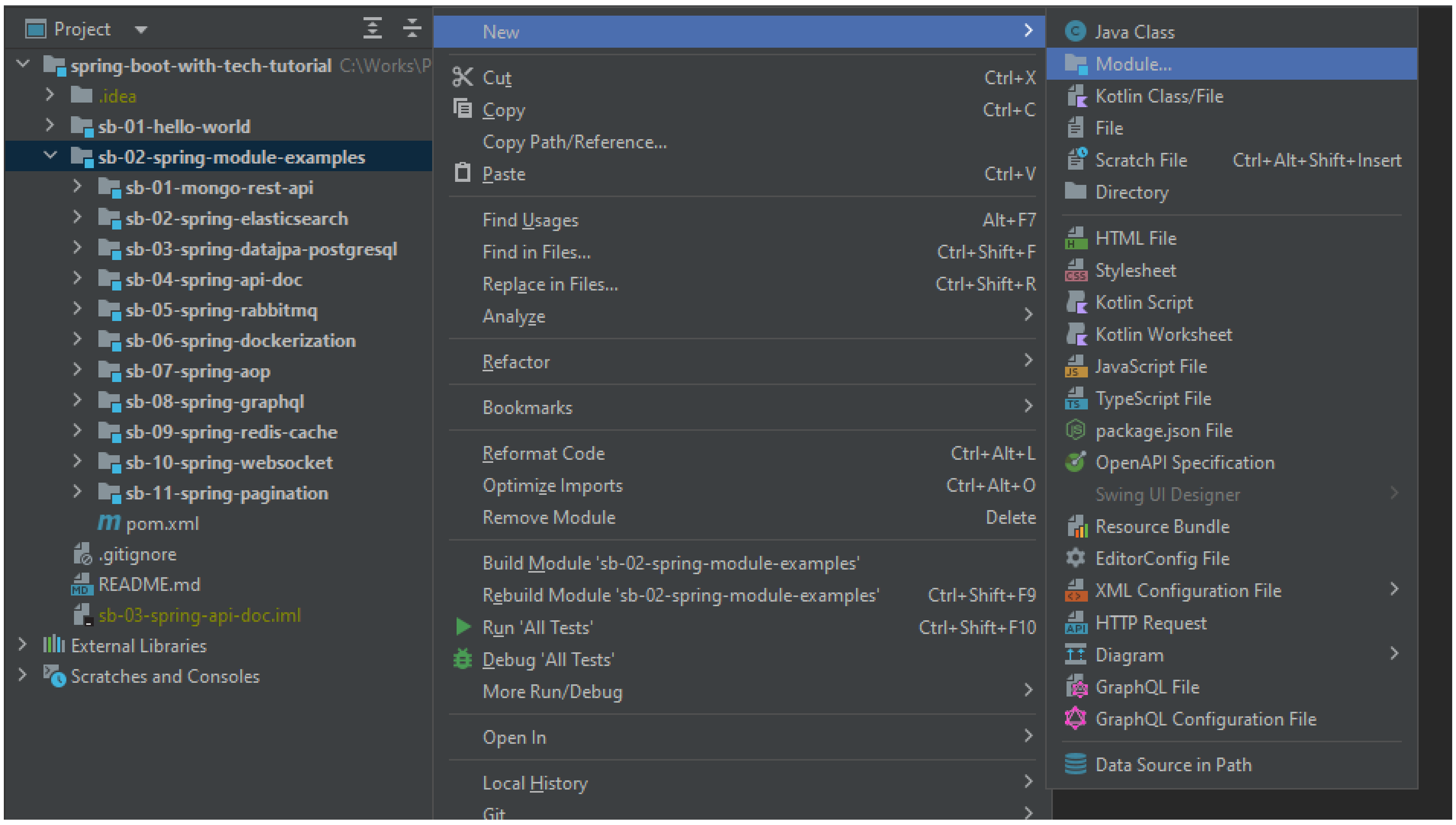
Event Driven Application

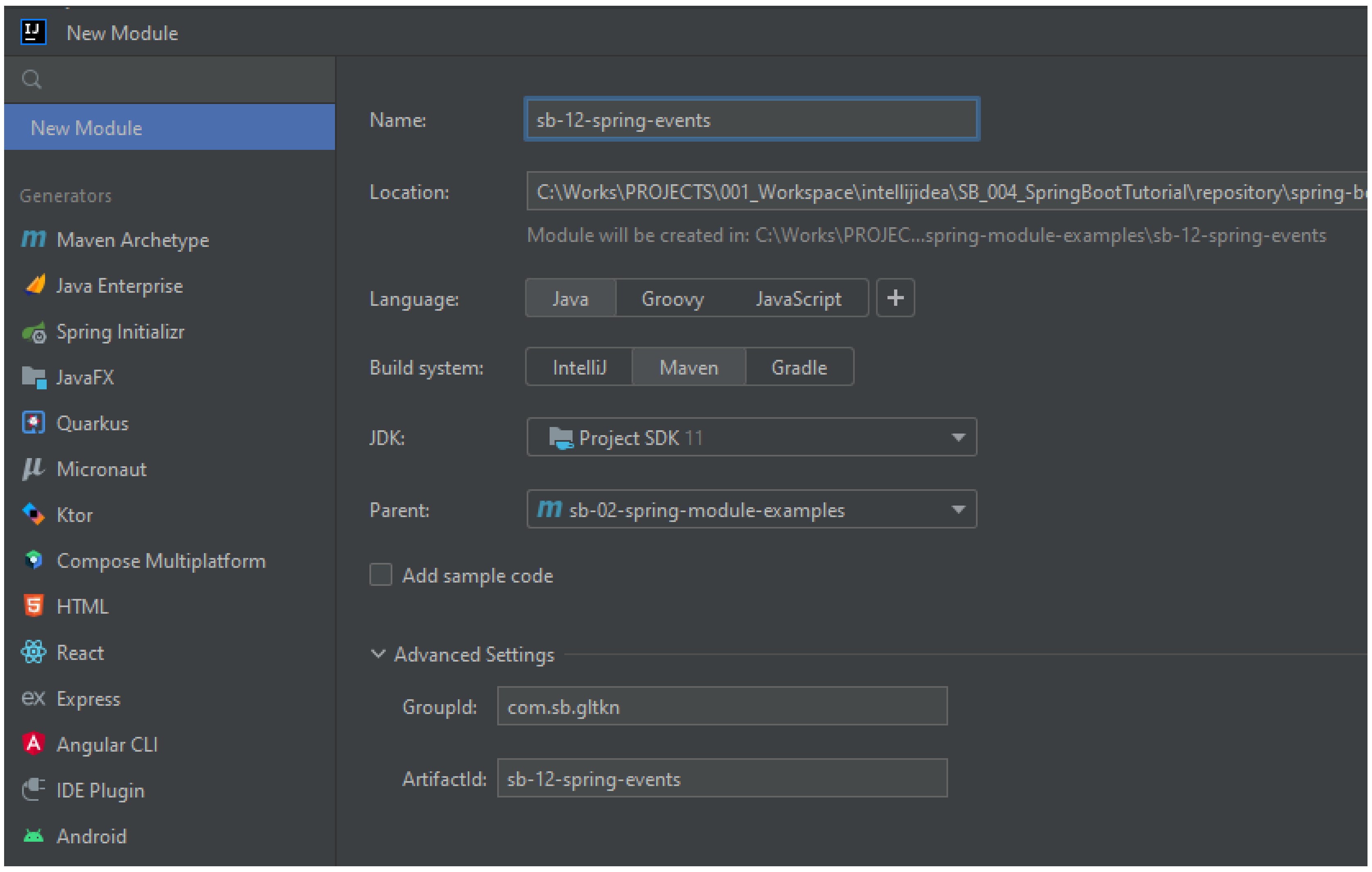
Event bazlı uygulama yazmak için ayrıca bir dependency eklememiz gerek yoktur. Spring'in bulundurduğu ***ApplicationEvent*** 'i implemente ederek sağlamaktayız.

Aşağıdaki örneği uygulayalım.



Modülümüzü oluşturalım.





Name: sb-12-spring-events

Location: C:\Works\PROJECTS\001_Workspace\intellijidea\SB_004_SpringBootTutorial\repository\spring-b

Module will be created in: C:\Works\PROJEC...spring-module-examples\sb-12-spring-events

Language: [✓ Java] Groovy JavaScript +

Build system: IntelliJ [✓ Maven] Gradle

JDK: Project SDK 11

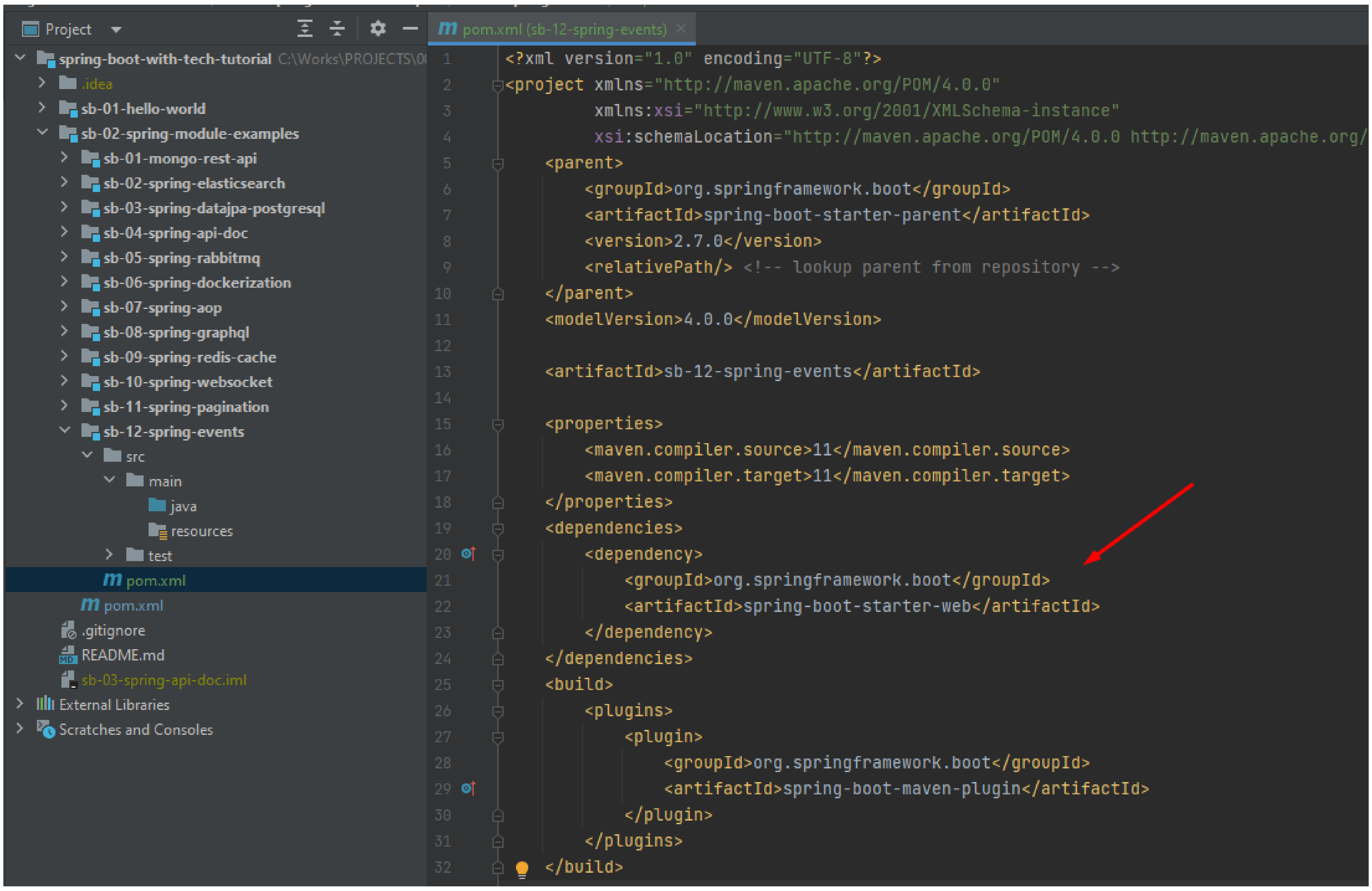
Parent: sb-02-spring-module-examples

☐ Add sample code

Advanced Settings

GroupId: com.sb.gltkn

ArtifactId: sb-12-spring-events

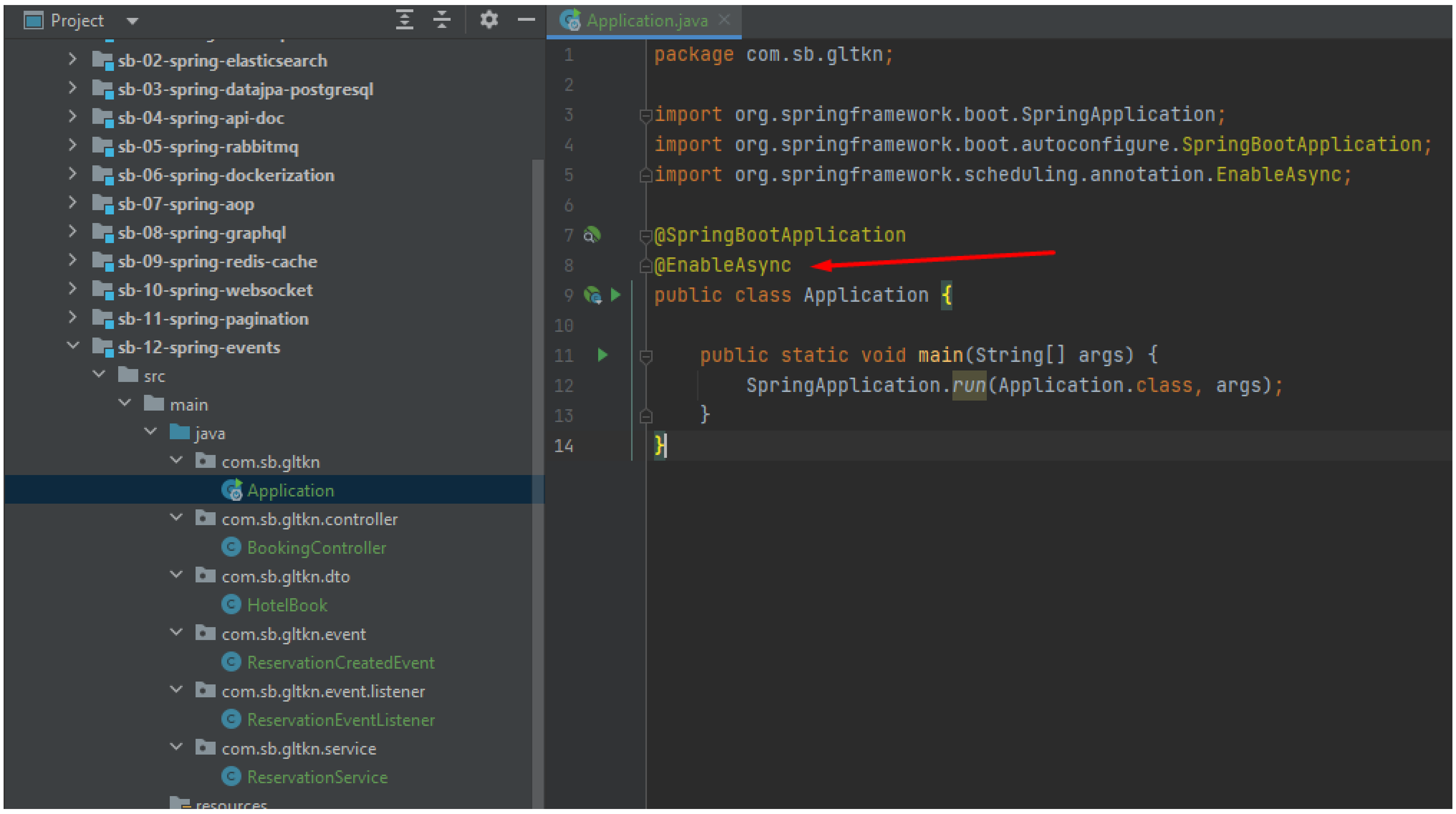


```
Project
spring-boot-with-tech-tutorial C:\Works\PROJECTS\0
  .idea
  sb-01-hello-world
  sb-02-spring-module-examples
    sb-01-mongo-rest-api
    sb-02-spring-elasticsearch
    sb-03-spring-datajpa-postgresql
    sb-04-spring-api-doc
    sb-05-spring-rabbitmq
    sb-06-spring-dockerization
    sb-07-spring-aop
    sb-08-spring-graphql
    sb-09-spring-redis-cache
    sb-10-spring-websocket
    sb-11-spring-pagination
    sb-12-spring-events
      src
        main
          java
          resources
        test
      pom.xml
    pom.xml
  .gitignore
  README.md
  sb-03-spring-api-doc.iml
External Libraries
Scratches and Consoles
```

pom.xml (sb-12-spring-events)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>sb-12-spring-events</artifactId>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

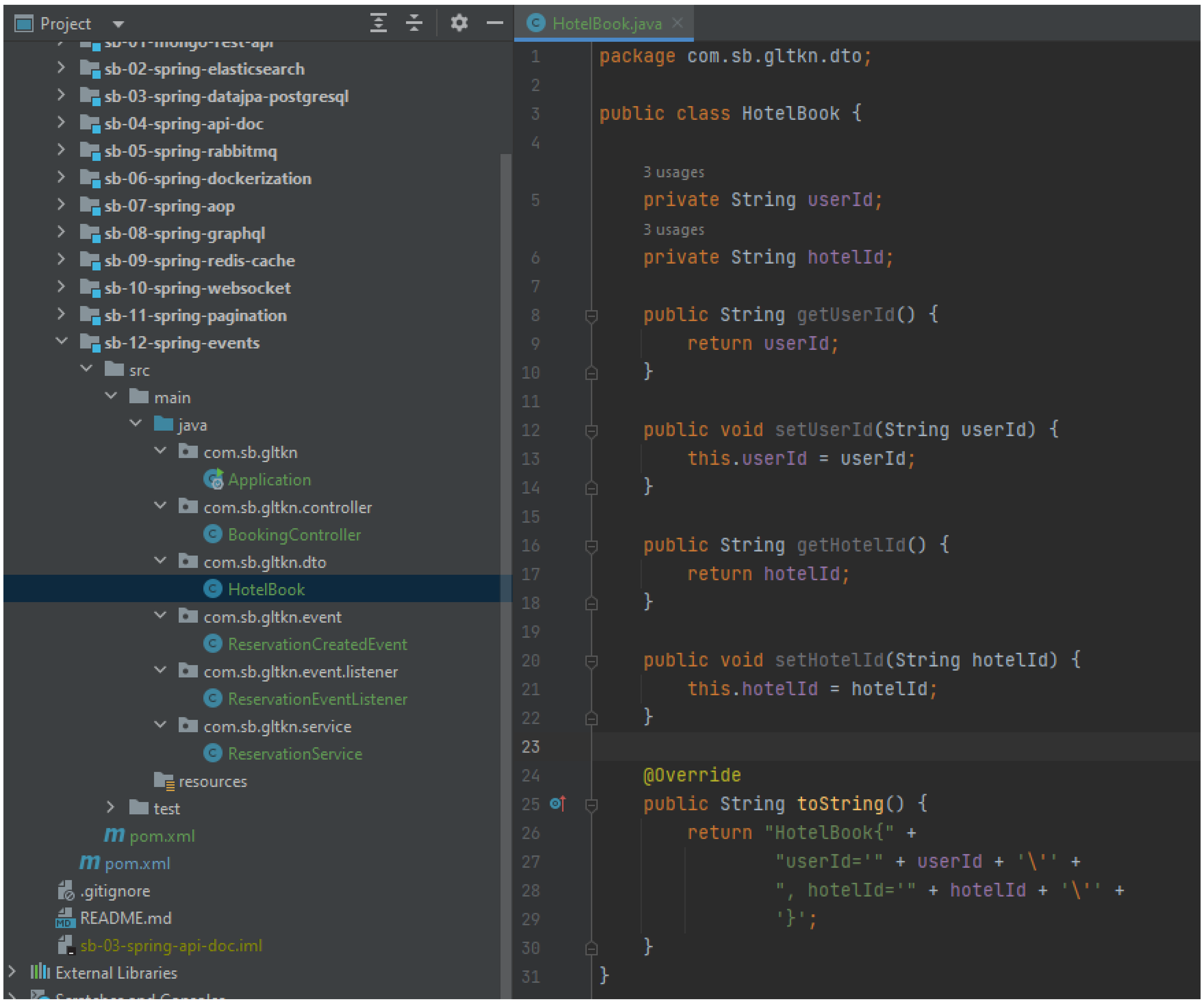
SpringBoot context'i içerisinde asenkron çalışmayı enable edelim.



```
Project
    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
    sb-08-spring-graphql
    sb-09-spring-redis-cache
    sb-10-spring-websocket
    sb-11-spring-pagination
    sb-12-spring-events
      src
        main
          java
            com.sb.gltkn
              Application
            com.sb.gltkn.controller
              BookingController
            com.sb.gltkn.dto
              HotelBook
            com.sb.gltkn.event
              ReservationCreatedEvent
            com.sb.gltkn.event.listener
              ReservationEventListener
            com.sb.gltkn.service
              ReservationService
          resources
```

Application.java

```java
package com.sb.gltkn;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableAsync;

@SpringBootApplication
@EnableAsync
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

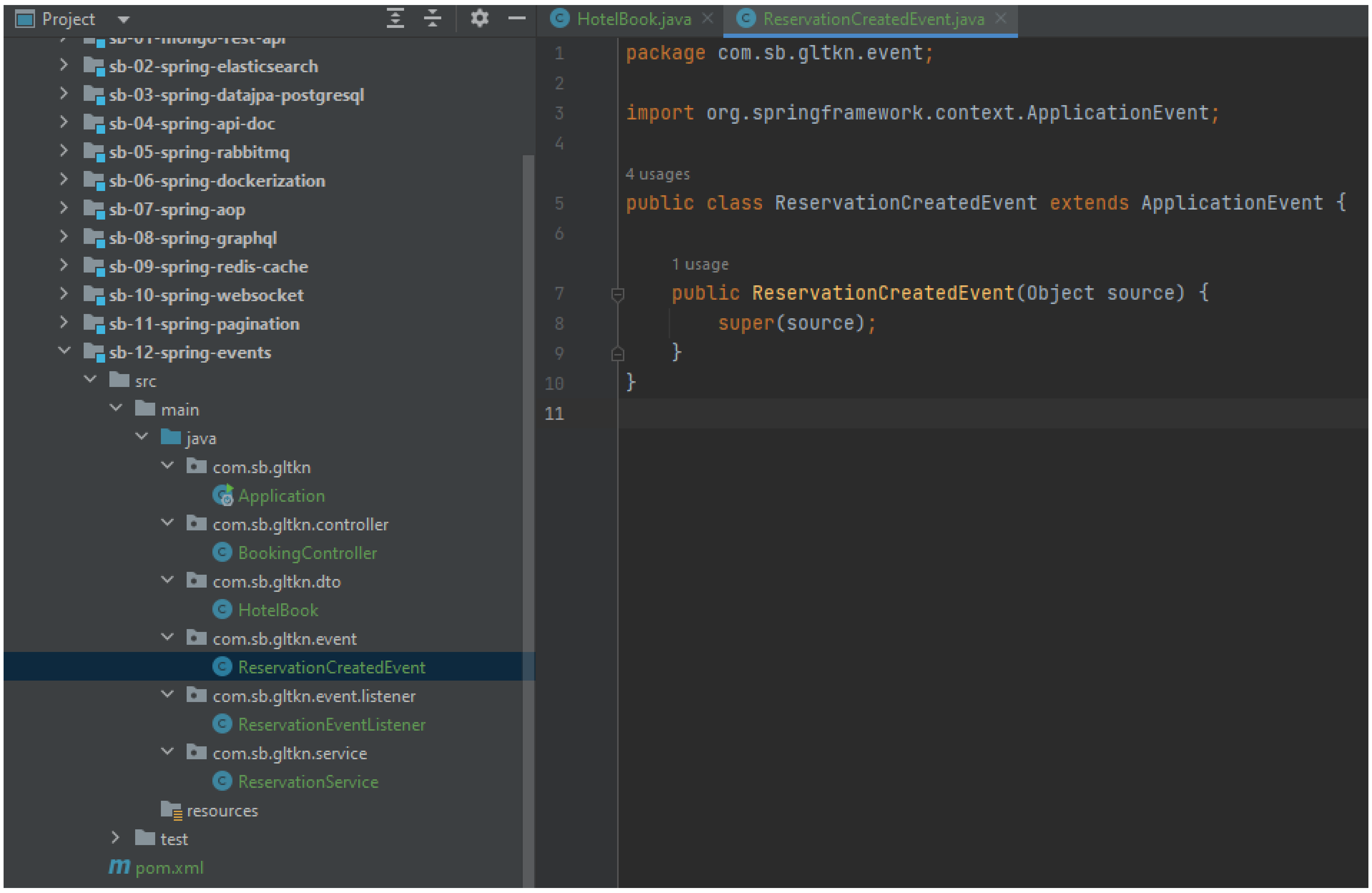


Project

```
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
> sb-08-spring-graphql
> sb-09-spring-redis-cache
> sb-10-spring-websocket
> sb-11-spring-pagination
v sb-12-spring-events
  v src
    v main
      v java
        v com.sb.gltkn
            Application
        v com.sb.gltkn.controller
            BookingController
        v com.sb.gltkn.dto
            HotelBook
        v com.sb.gltkn.event
            ReservationCreatedEvent
        v com.sb.gltkn.event.listener
            ReservationEventListener
        v com.sb.gltkn.service
            ReservationService
        resources
    > test
    pom.xml
  pom.xml
  .gitignore
  README.md
  sb-03-spring-api-doc.iml
> External Libraries
> Scratches and Consoles
```

HotelBook.java

```java
package com.sb.gltkn.dto;

public class HotelBook {

    3 usages
    private String userId;
    3 usages
    private String hotelId;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getHotelId() {
        return hotelId;
    }

    public void setHotelId(String hotelId) {
        this.hotelId = hotelId;
    }

    @Override
    public String toString() {
        return "HotelBook{" +
                "userId='" + userId + '\'' +
                ", hotelId='" + hotelId + '\'' +
                '}';
    }
}
```



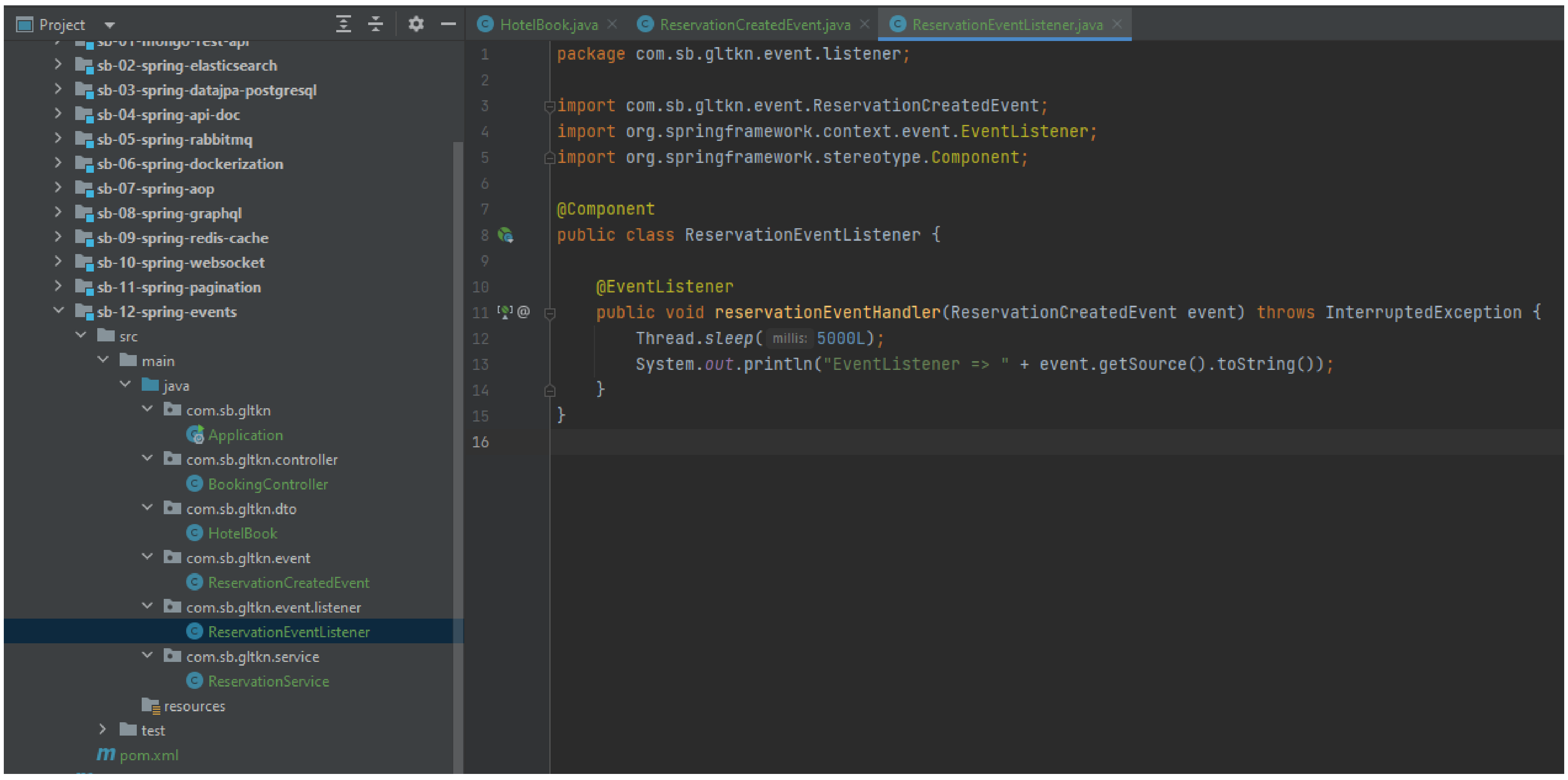
Project

```
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
> sb-08-spring-graphql
> sb-09-spring-redis-cache
> sb-10-spring-websocket
> sb-11-spring-pagination
v sb-12-spring-events
  v src
    v main
      v java
        v com.sb.gltkn
            Application
        v com.sb.gltkn.controller
            BookingController
        v com.sb.gltkn.dto
            HotelBook
        v com.sb.gltkn.event
            ReservationCreatedEvent
        v com.sb.gltkn.event.listener
            ReservationEventListener
        v com.sb.gltkn.service
            ReservationService
        resources
    > test
    pom.xml
```

HotelBook.java   ReservationCreatedEvent.java

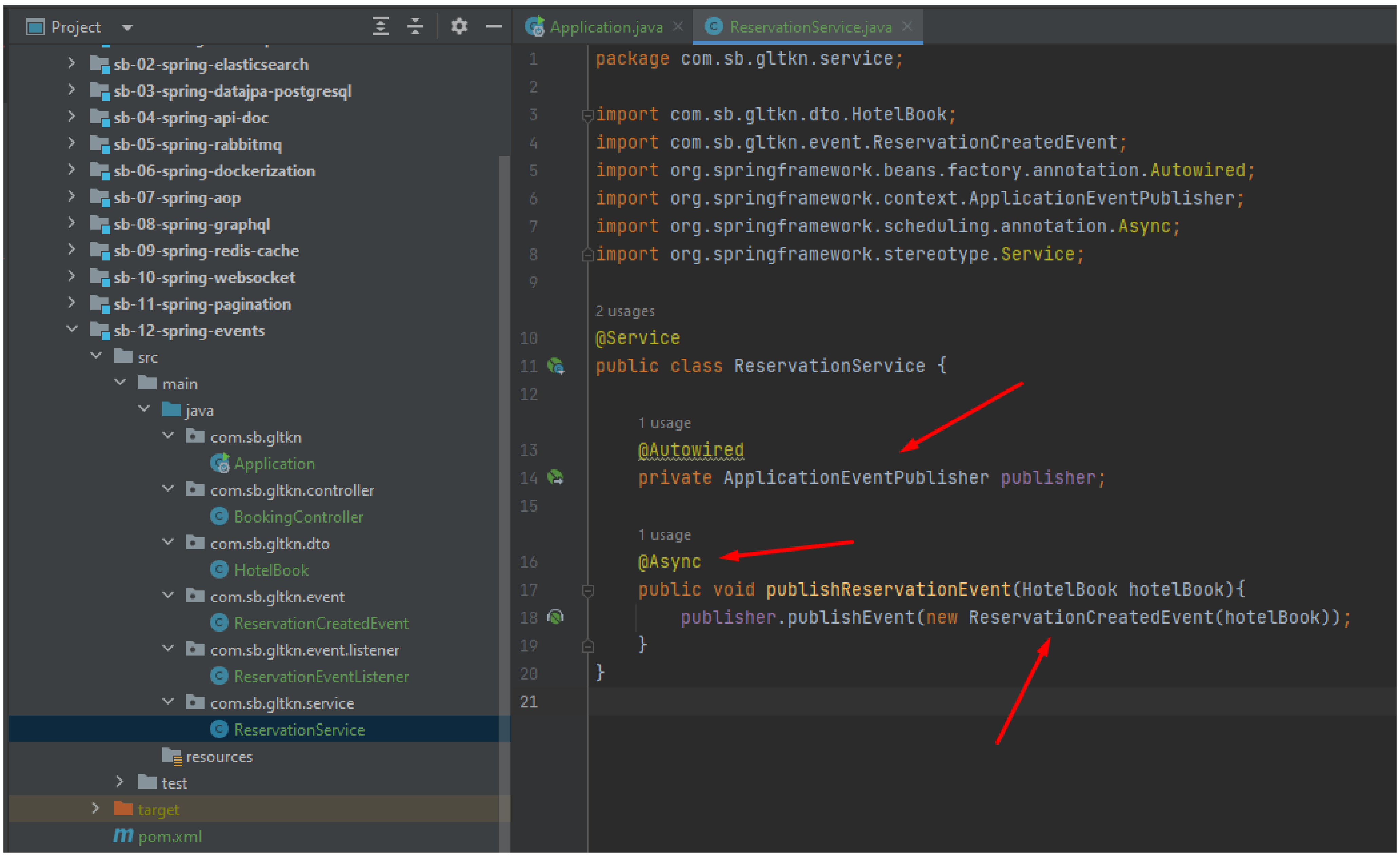
```java
package com.sb.gltkn.event;

import org.springframework.context.ApplicationEvent;

4 usages
public class ReservationCreatedEvent extends ApplicationEvent {

    1 usage
    public ReservationCreatedEvent(Object source) {
        super(source);
    }
}
```

Aşağıdaki class'ta publish edilen event'ı dinliyoruz.
Burada aslında üstte oluşturduğumuz **ApplicationEvent**'tan türeyen ***ReservationCreatedEvent*** class'ını dinlemekteyiz.

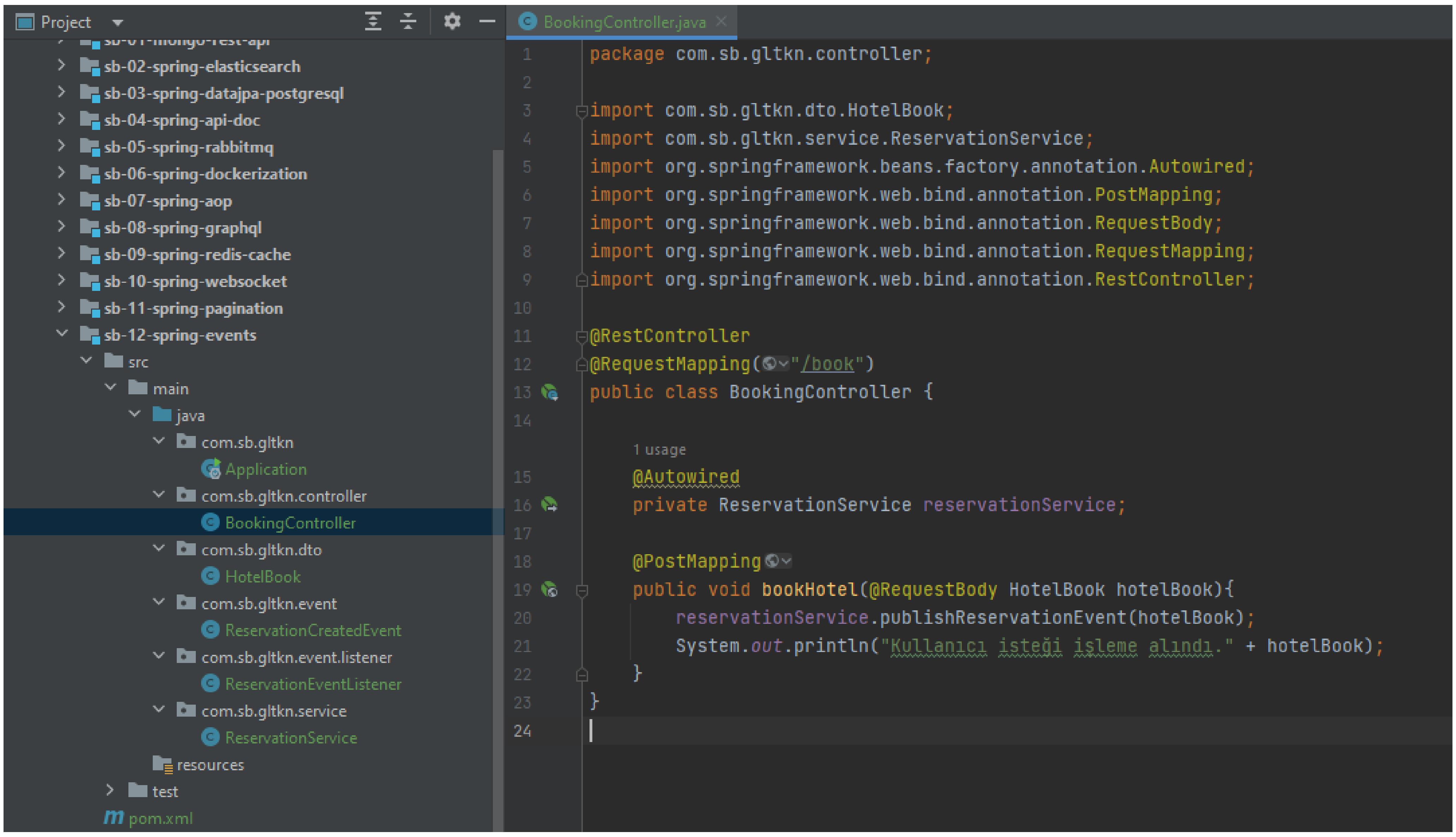


```java
package com.sb.gltkn.event.listener;

import com.sb.gltkn.event.ReservationCreatedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

@Component
public class ReservationEventListener {

    @EventListener
    public void reservationEventHandler(ReservationCreatedEvent event) throws InterruptedException {
        Thread.sleep( millis: 5000L);
        System.out.println("EventListener => " + event.getSource().toString());
    }
}
```

Servis class'ımızda ise event'ı publish ediyoruz.
Publish ettiğimiz ApplicationEvent'ımız ise üstte oluşturduğumuz ***ReservationCreatedEvent*** class'ıdır. Bu class HotelBook nesnesini almaktadır.



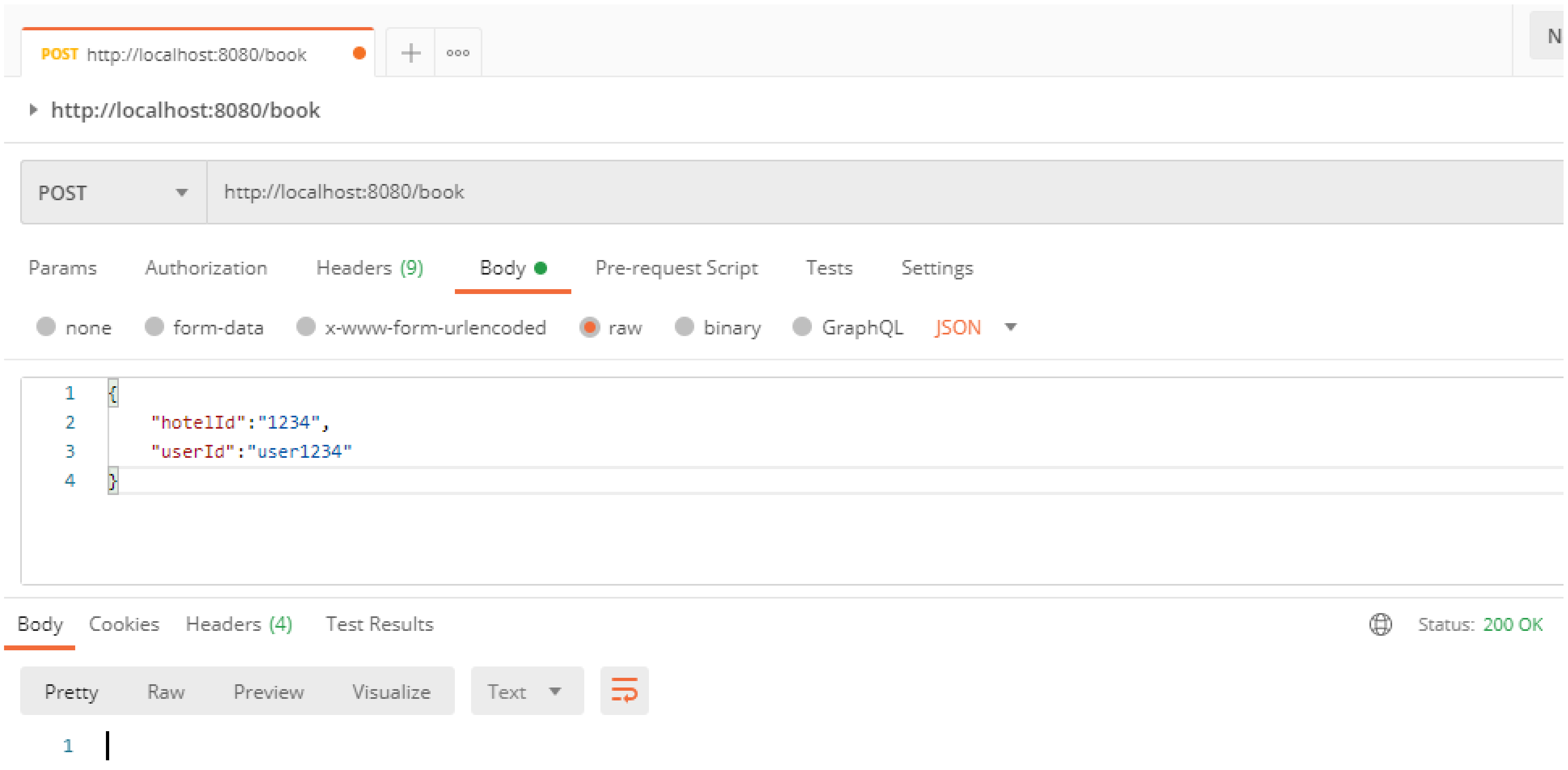
```java
package com.sb.gltkn.service;

import com.sb.gltkn.dto.HotelBook;
import com.sb.gltkn.event.ReservationCreatedEvent;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationEventPublisher;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

2 usages
@Service
public class ReservationService {

    1 usage
    @Autowired
    private ApplicationEventPublisher publisher;

    1 usage
    @Async
    public void publishReservationEvent(HotelBook hotelBook){
        publisher.publishEvent(new ReservationCreatedEvent(hotelBook));
    }
}
```



```java
package com.sb.gltkn.controller;

import com.sb.gltkn.dto.HotelBook;
import com.sb.gltkn.service.ReservationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/book")
public class BookingController {

    1 usage
    @Autowired
    private ReservationService reservationService;

    @PostMapping
    public void bookHotel(@RequestBody HotelBook hotelBook){
        reservationService.publishReservationEvent(hotelBook);
        System.out.println("Kullanıcı isteği işleme alındı." + hotelBook);
    }
}
```

İsteğimizi yaptığımızda asenkron olarak event handler çalışacaktır.
handle esnasında 5 saniye beklettik. Bu süre içerisinde işlemlerin gerçekleştiğini düşünelim. Örneğin notification işlemi sağlanıyor vs. gibi.

```json
{
    "hotelId":"1234",
    "userId":"user1234"
}
```



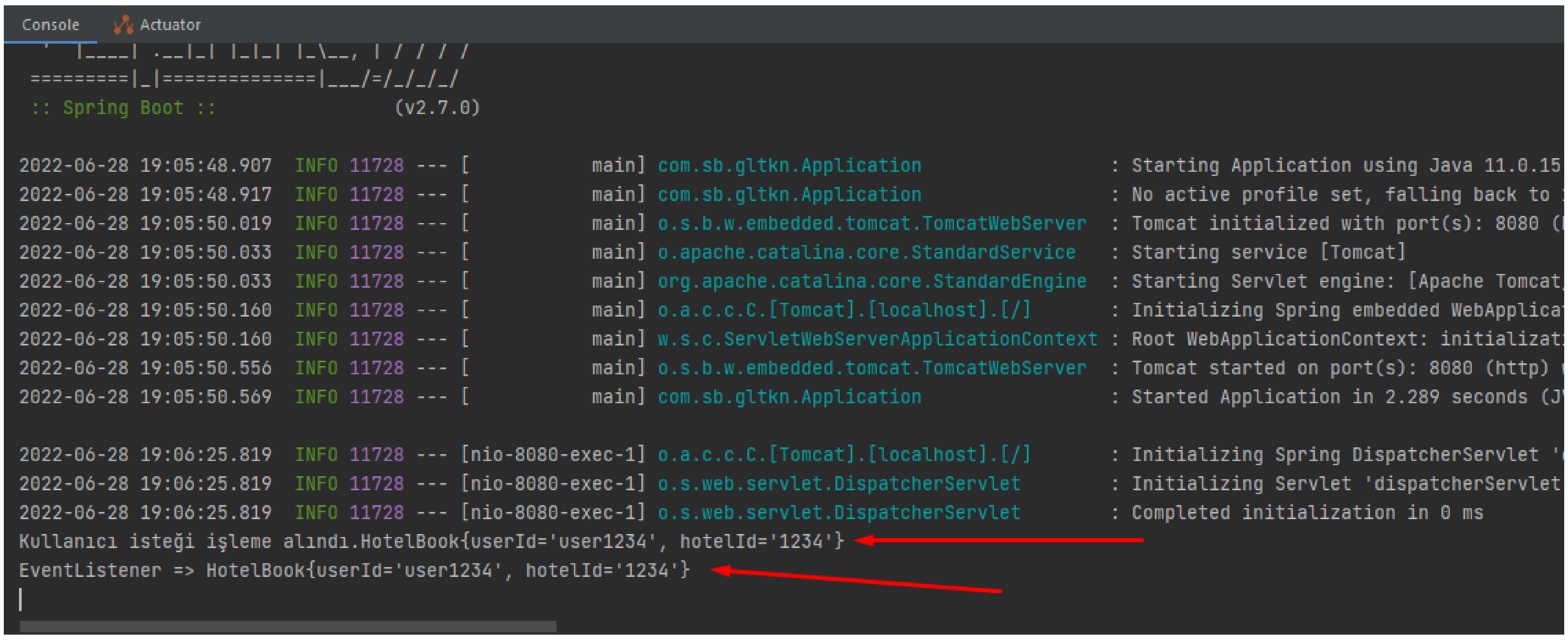
POST http://localhost:8080/book

Body: [✓ raw] JSON

```json
{
    "hotelId":"1234",
    "userId":"user1234"
}
```

Status: 200 OK

Alttaki ikinci ok ile gösterilen log 5 saniye sonra düşmüştür.



```
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.0)

2022-06-28 19:05:48.907  INFO 11728 --- [           main] com.sb.gltkn.Application                 : Starting Application using Java 11.0.15
2022-06-28 19:05:48.917  INFO 11728 --- [           main] com.sb.gltkn.Application                 : No active profile set, falling back to
2022-06-28 19:05:50.019  INFO 11728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (
2022-06-28 19:05:50.033  INFO 11728 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-06-28 19:05:50.033  INFO 11728 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat
2022-06-28 19:05:50.160  INFO 11728 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplica
2022-06-28 19:05:50.160  INFO 11728 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializat
2022-06-28 19:05:50.556  INFO 11728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
2022-06-28 19:05:50.569  INFO 11728 --- [           main] com.sb.gltkn.Application                 : Started Application in 2.289 seconds (J

2022-06-28 19:06:25.819  INFO 11728 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet '
2022-06-28 19:06:25.819  INFO 11728 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet
2022-06-28 19:06:25.819  INFO 11728 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 0 ms
Kullanıcı isteği işleme alındı.HotelBook{userId='user1234', hotelId='1234'}
EventListener => HotelBook{userId='user1234', hotelId='1234'}
```