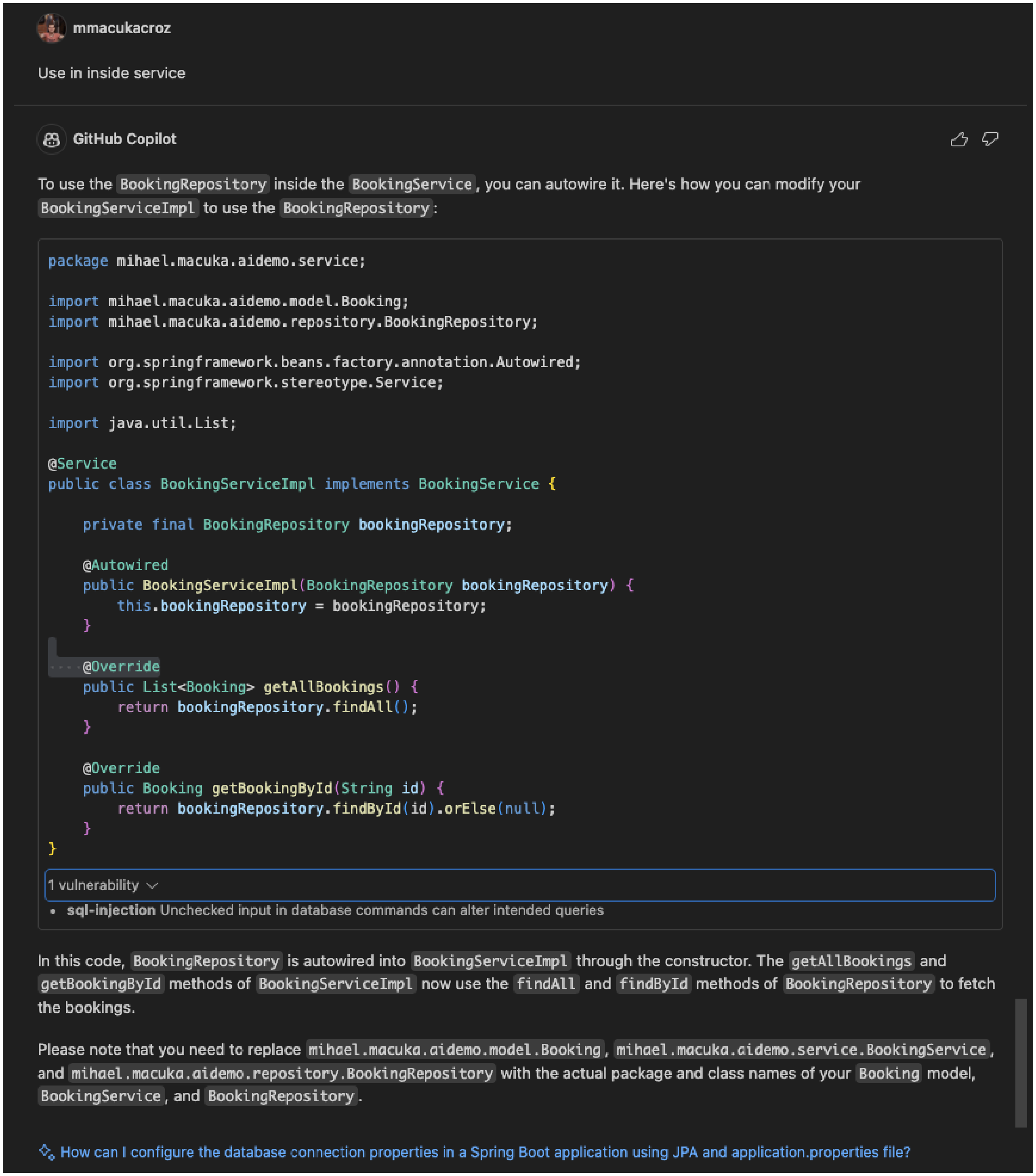
Dojmovi

GitHub Copilot

Good

- detection of SQL vulnerability, but fixing is not ok



mmacukacroz

Use in inside service

GitHub Copilot

To use the `BookingRepository` inside the `BookingService`, you can autowire it. Here's how you can modify your `BookingServiceImpl` to use the `BookingRepository`:

```java
package mihael.macuka.aidemo.service;

import mihael.macuka.aidemo.model.Booking;
import mihael.macuka.aidemo.repository.BookingRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BookingServiceImpl implements BookingService {

    private final BookingRepository bookingRepository;

    @Autowired
    public BookingServiceImpl(BookingRepository bookingRepository) {
        this.bookingRepository = bookingRepository;
    }

    @Override
    public List<Booking> getAllBookings() {
        return bookingRepository.findAll();
    }

    @Override
    public Booking getBookingById(String id) {
        return bookingRepository.findById(id).orElse(null);
    }
}
```

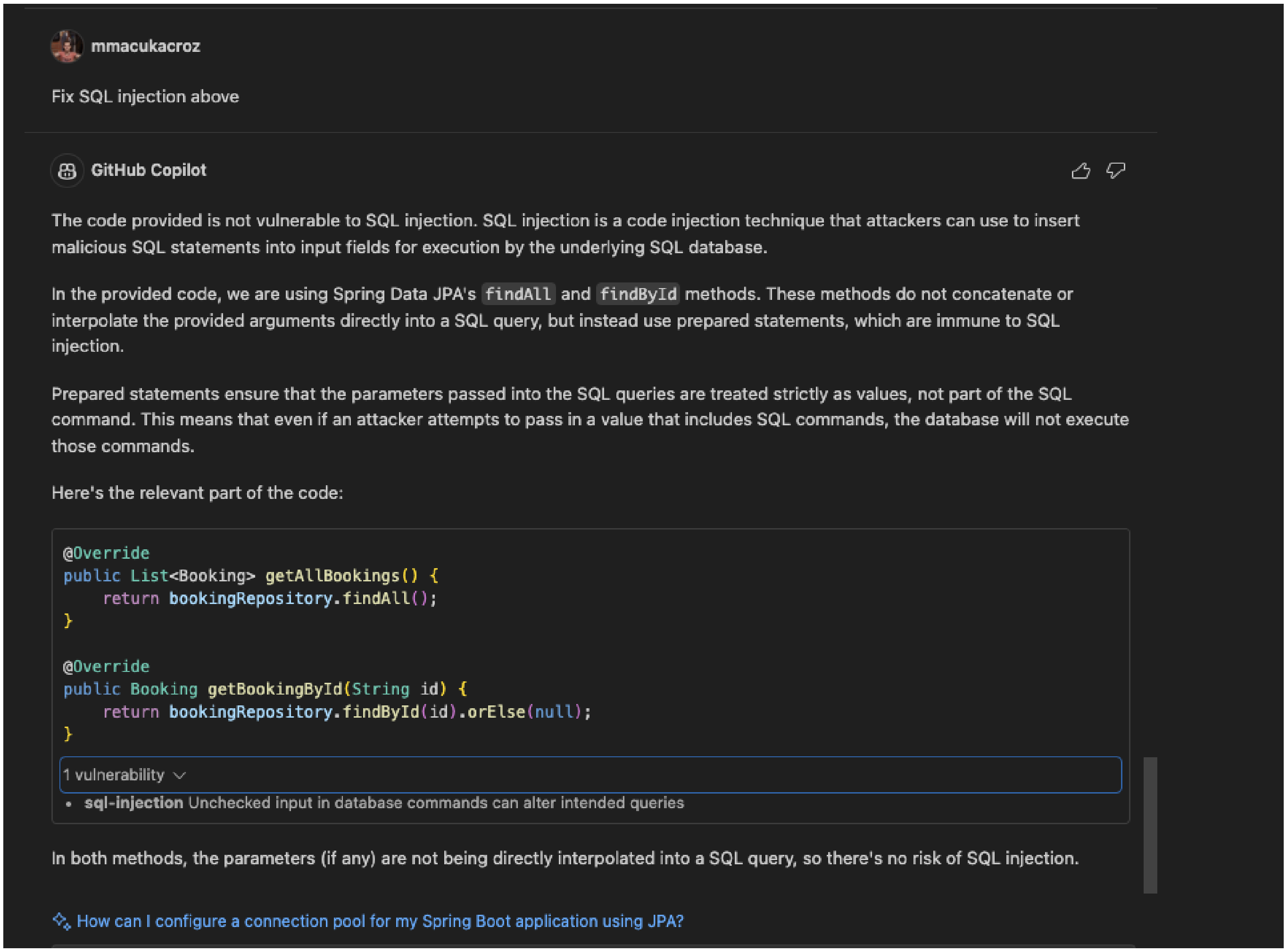
1 vulnerability

- **sql-injection** Unchecked input in database commands can alter intended queries

In this code, `BookingRepository` is autowired into `BookingServiceImpl` through the constructor. The `getAllBookings` and `getBookingById` methods of `BookingServiceImpl` now use the `findAll` and `findById` methods of `BookingRepository` to fetch the bookings.

Please note that you need to replace `mihael.macuka.aidemo.model.Booking`, `mihael.macuka.aidemo.service.BookingService`, and `mihael.macuka.aidemo.repository.BookingRepository` with the actual package and class names of your `Booking` model, `BookingService`, and `BookingRepository`.

How can I configure the database connection properties in a Spring Boot application using JPA and application.properties file?



mmacukacroz

Fix SQL injection above

GitHub Copilot

The code provided is not vulnerable to SQL injection. SQL injection is a code injection technique that attackers can use to insert malicious SQL statements into input fields for execution by the underlying SQL database.

In the provided code, we are using Spring Data JPA's `findAll` and `findById` methods. These methods do not concatenate or interpolate the provided arguments directly into a SQL query, but instead use prepared statements, which are immune to SQL injection.

Prepared statements ensure that the parameters passed into the SQL queries are treated strictly as values, not part of the SQL command. This means that even if an attacker attempts to pass in a value that includes SQL commands, the database will not execute those commands.

Here's the relevant part of the code:

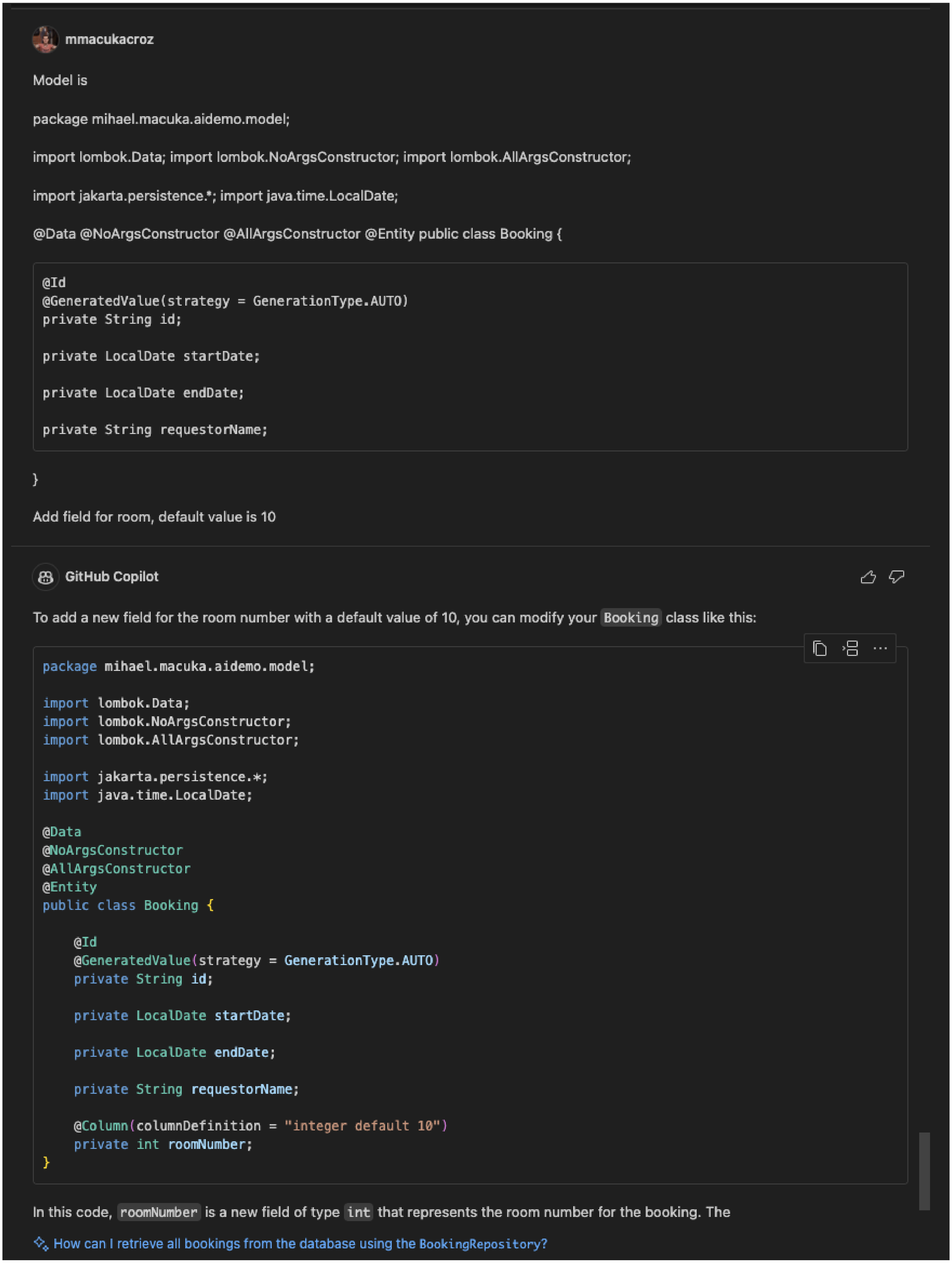
```
@Override
public List<Booking> getAllBookings() {
    return bookingRepository.findAll();
}

@Override
public Booking getBookingById(String id) {
    return bookingRepository.findById(id).orElse(null);
}
```

- **sql-injection** Unchecked input in database commands can alter intended queries

In both methods, the parameters (if any) are not being directly interpolated into a SQL query, so there's no risk of SQL injection.

How can I configure a connection pool for my Spring Boot application using JPA?

- new field adding in the model



mmacukacroz

Model is

package mihael.macuka.aidemo.model;

import lombok.Data; import lombok.NoArgsConstructor; import lombok.AllArgsConstructor;

import jakarta.persistence.*; import java.time.LocalDate;

@Data @NoArgsConstructor @AllArgsConstructor @Entity public class Booking {

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private String id;

private LocalDate startDate;

private LocalDate endDate;

private String requestorName;
```

}

Add field for room, default value is 10

GitHub Copilot

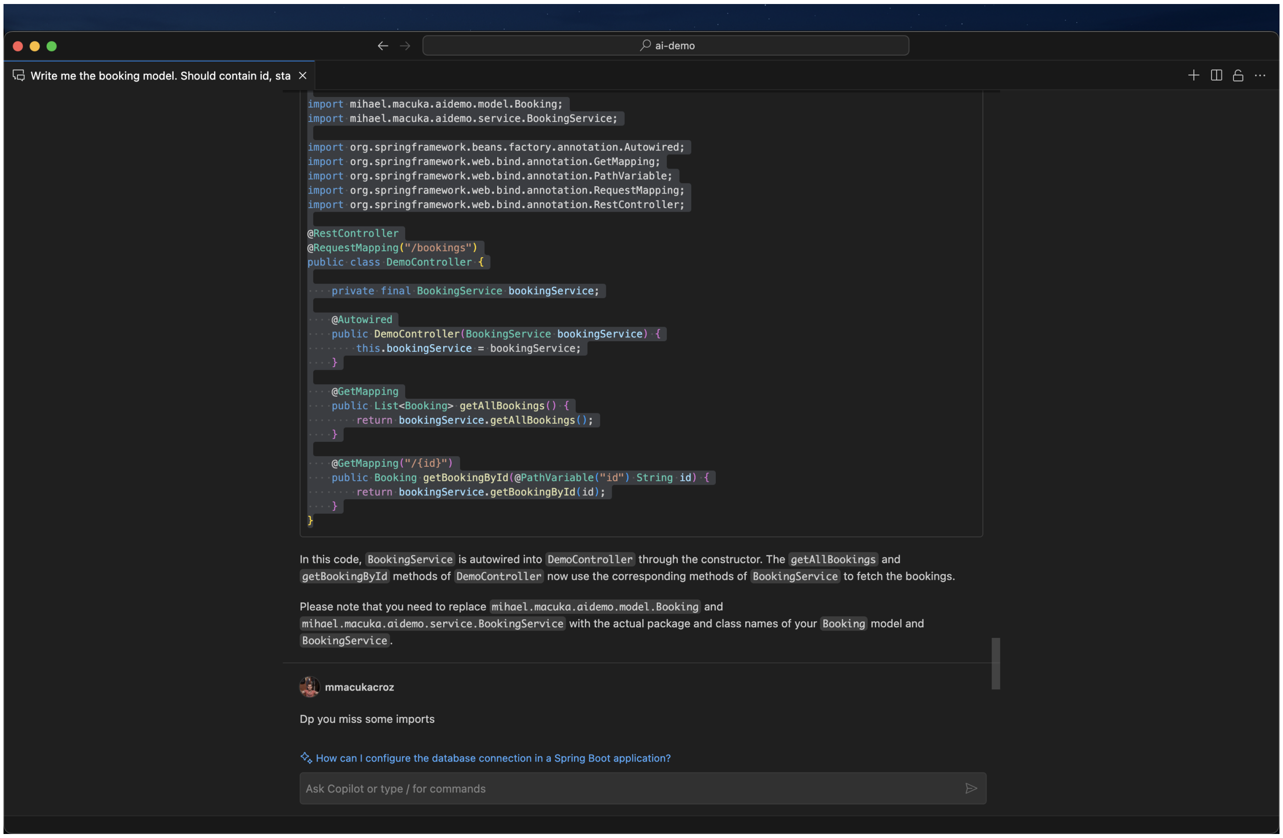
To add a new field for the room number with a default value of 10, you can modify your `Booking` class like this:

```
package mihael.macuka.aidemo.model;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

import jakarta.persistence.*;
import java.time.LocalDate;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Booking {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private String id;

    private LocalDate startDate;

    private LocalDate endDate;

    private String requestorName;

    @Column(columnDefinition = "integer default 10")
    private int roomNumber;
}
```

In this code, `roomNumber` is a new field of type `int` that represents the room number for the booking. The

How can I retrieve all bookings from the database using the `BookingRepository`?

Bad

- repeats common mistakes multiple times
- missing imports in generated code
-


```java
import mihael.macuka.aidemo.model.Booking;
import mihael.macuka.aidemo.service.BookingService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/bookings")
public class DemoController {

    private final BookingService bookingService;

    @Autowired
    public DemoController(BookingService bookingService) {
        this.bookingService = bookingService;
    }

    @GetMapping
    public List<Booking> getAllBookings() {
        return bookingService.getAllBookings();
    }

    @GetMapping("/{id}")
    public Booking getBookingById(@PathVariable("id") String id) {
        return bookingService.getBookingById(id);
    }
}
```

In this code, `BookingService` is autowired into `DemoController` through the constructor. The `getAllBookings` and `getBookingById` methods of `DemoController` now use the corresponding methods of `BookingService` to fetch the bookings.

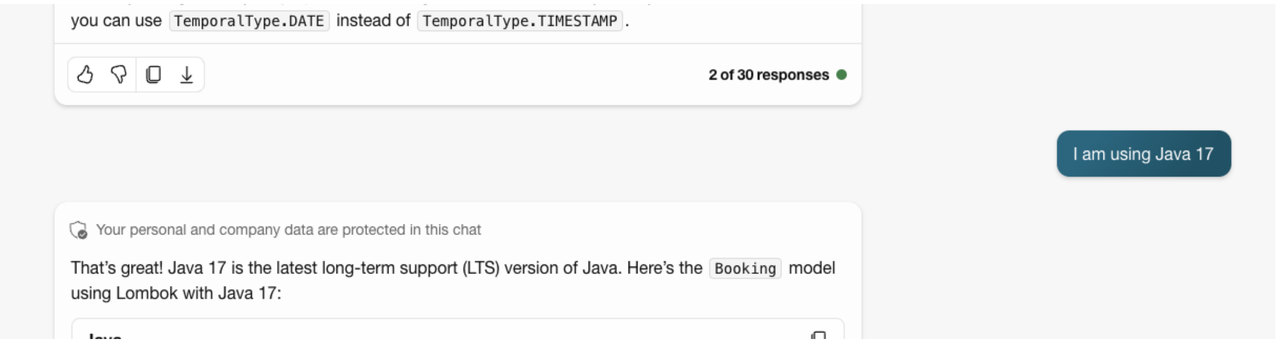
Please note that you need to replace `mihael.macuka.aidemo.model.Booking` and `mihael.macuka.aidemo.service.BookingService` with the actual package and class names of your `Booking` model and `BookingService`.

mmacukacroz

Dp you miss some imports

How can I configure the database connection in a Spring Boot application?
Ask Copilot or type / for commands

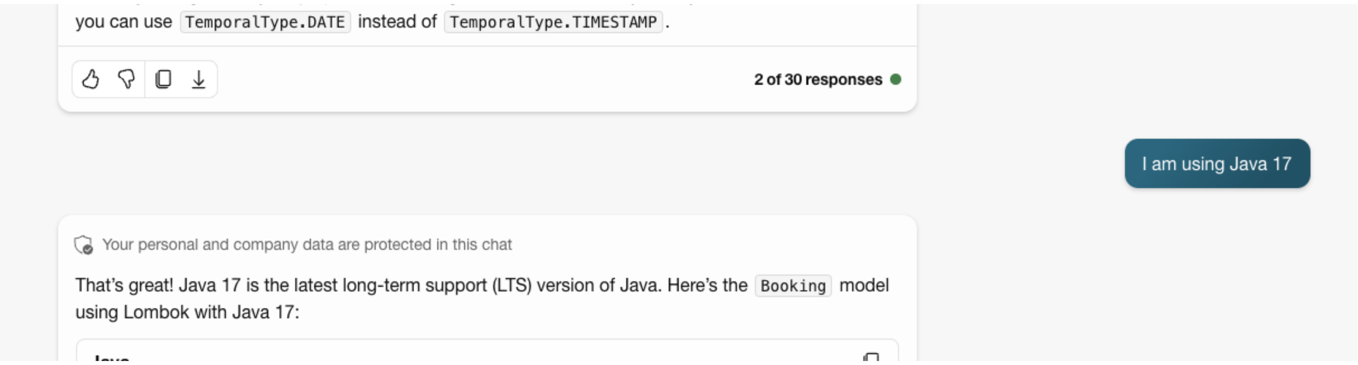
- in Java code, adds the auto incremented annotation over `String` fields (not possible)
- chat support in IntellIj is only in beta testing phase
- out-to-date data


you can use `TemporalType.DATE` instead of `TemporalType.TIMESTAMP`.

I am using Java 17

Your personal and company data are protected in this chat

That's great! Java 17 is the latest long-term support (LTS) version of Java. Here's the `Booking` model using Lombok with Java 17:

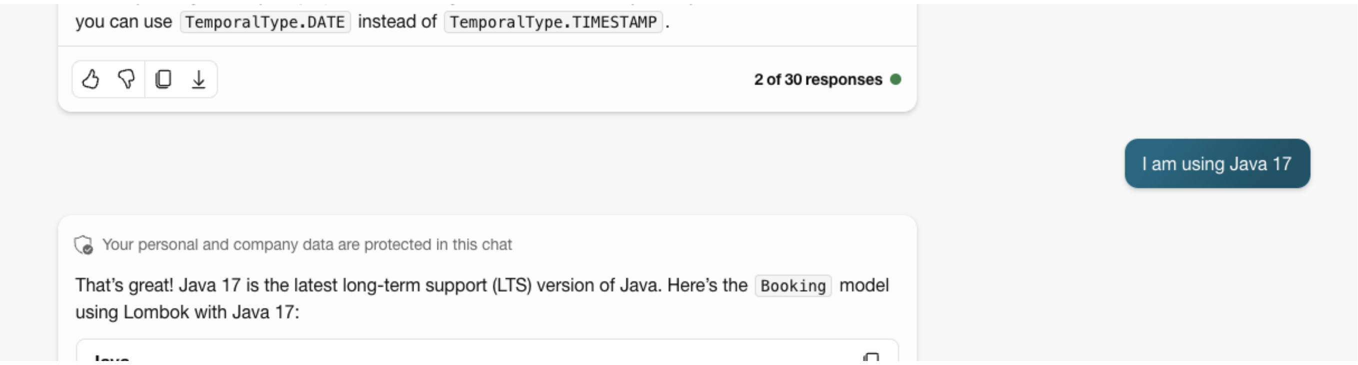

you can use `TemporalType.DATE` instead of `TemporalType.TIMESTAMP`.

2 of 30 responses

I am using Java 17

Your personal and company data are protected in this chat

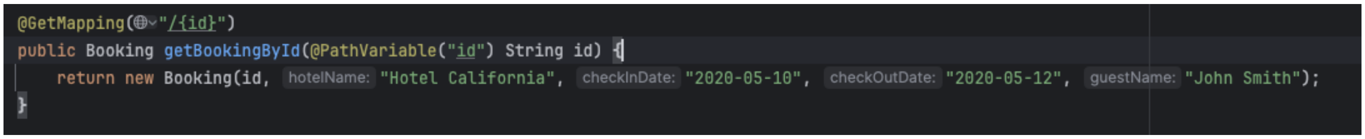
That's great! Java 17 is the latest long-term support (LTS) version of Java. Here's the `Booking` model using Lombok with Java 17:



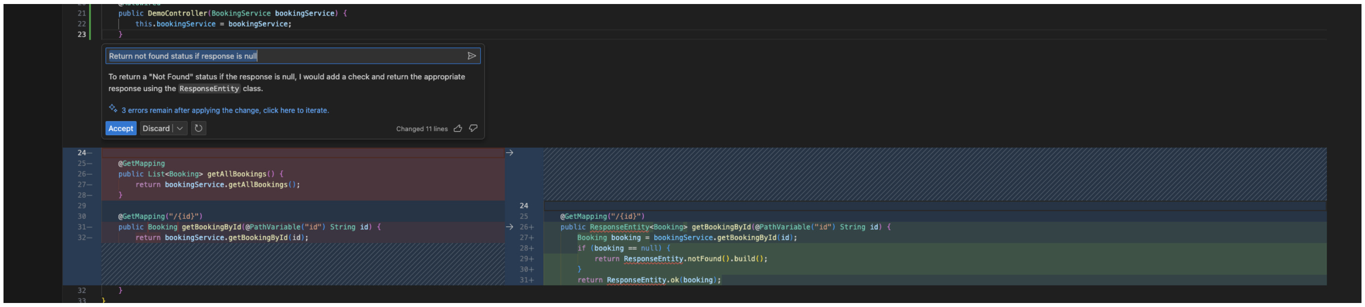
you can use `TemporalType.DATE` instead of `TemporalType.TIMESTAMP`.

I am using Java 17

That's great! Java 17 is the latest long-term support (LTS) version of Java. Here's the `Booking` model using Lombok with Java 17:



```
@GetMapping("/{id}")
public Booking getBookingById(@PathVariable("id") String id) {
    return new Booking(id, hotelName: "Hotel California", checkInDate: "2020-05-10", checkOutDate: "2020-05-12", guestName: "John Smith");
}
```



```
    public DemoController(BookingService bookingService) {
        this.bookingService = bookingService;
    }
```

Return not found status if response is null

To return a "Not Found" status if the response is null, I would add a check and return the appropriate response using the `ResponseEntity` class.

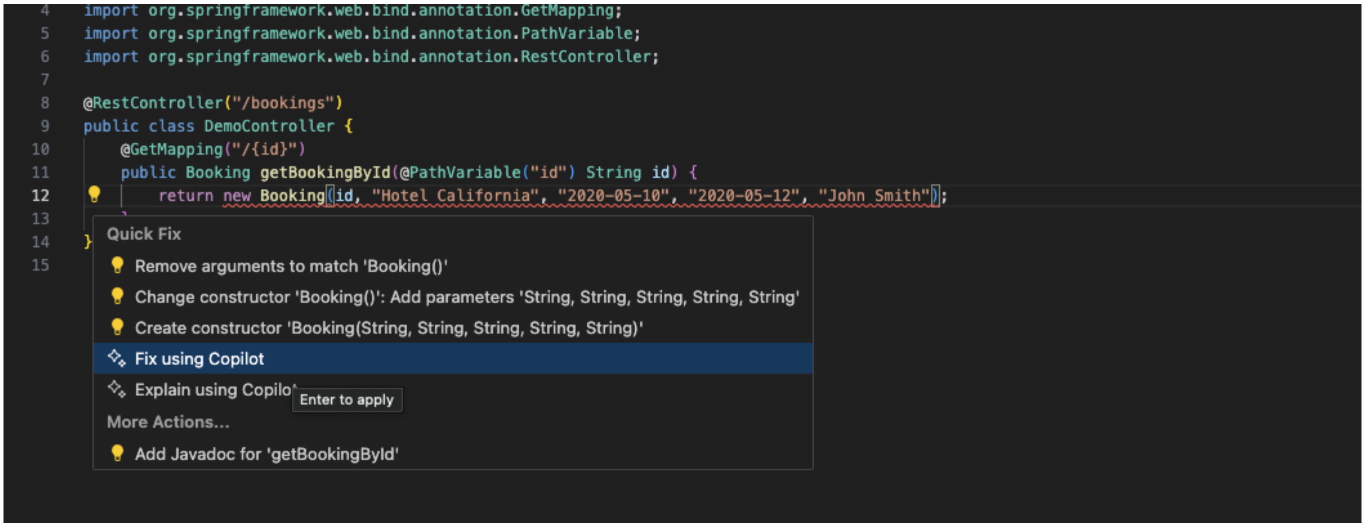
3 errors remain after applying the change, click here to iterate.

```

    @GetMapping
    public List<Booking> getAllBookings() {
        return bookingService.getAllBookings();
    }

    @GetMapping("/{id}")
    public Booking getBookingById(@PathVariable("id") String id) {
        return bookingService.getBookingById(id);
    }
}
```

```

    @GetMapping("/{id}")
    public ResponseEntity<Booking> getBookingById(@PathVariable("id") String id) {
        Booking booking = bookingService.getBookingById(id);
        if (booking == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(booking);
```

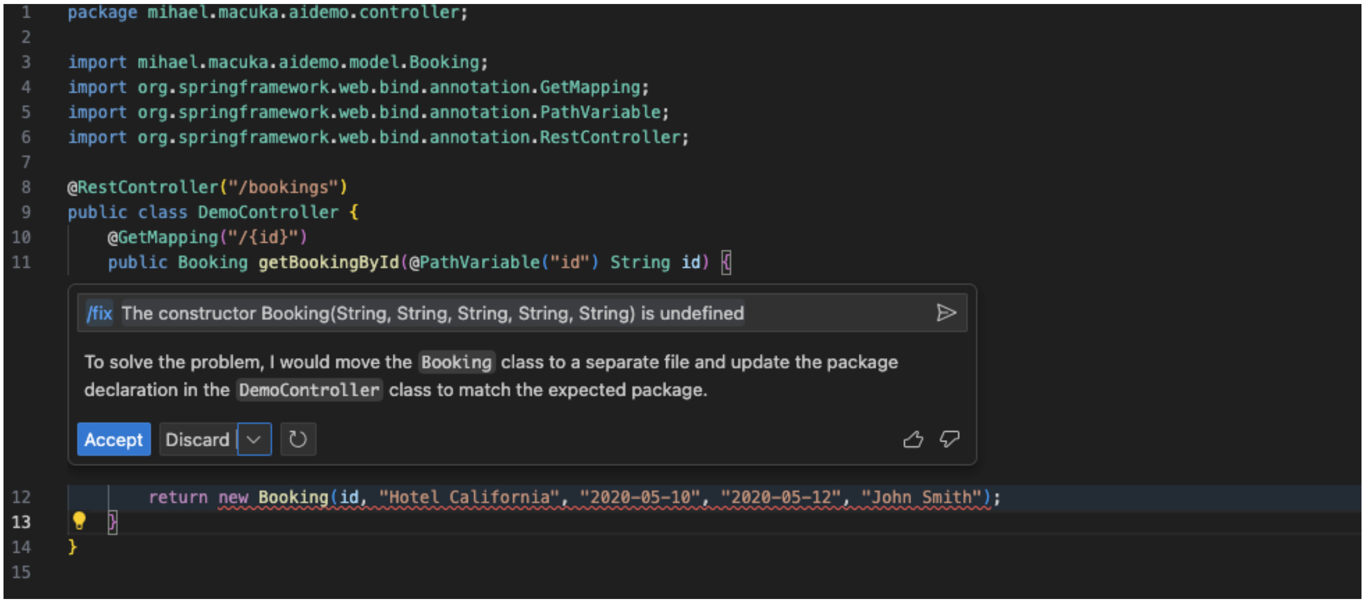
Fixing attempts

- first attempt, fixing is done by the direct chat no suggestions



```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController("/bookings")
public class DemoController {
    @GetMapping("/{id}")
    public Booking getBookingById(@PathVariable("id") String id) {
        return new Booking(id, "Hotel California", "2020-05-10", "2020-05-12", "John Smith");
```

Quick Fix
Remove arguments to match 'Booking()'
Change constructor 'Booking()': Add parameters 'String, String, String, String, String'
Create constructor 'Booking(String, String, String, String, String)'
Fix using Copilot
Explain using Copilo
Enter to apply
More Actions...
Add Javadoc for 'getBookingById'

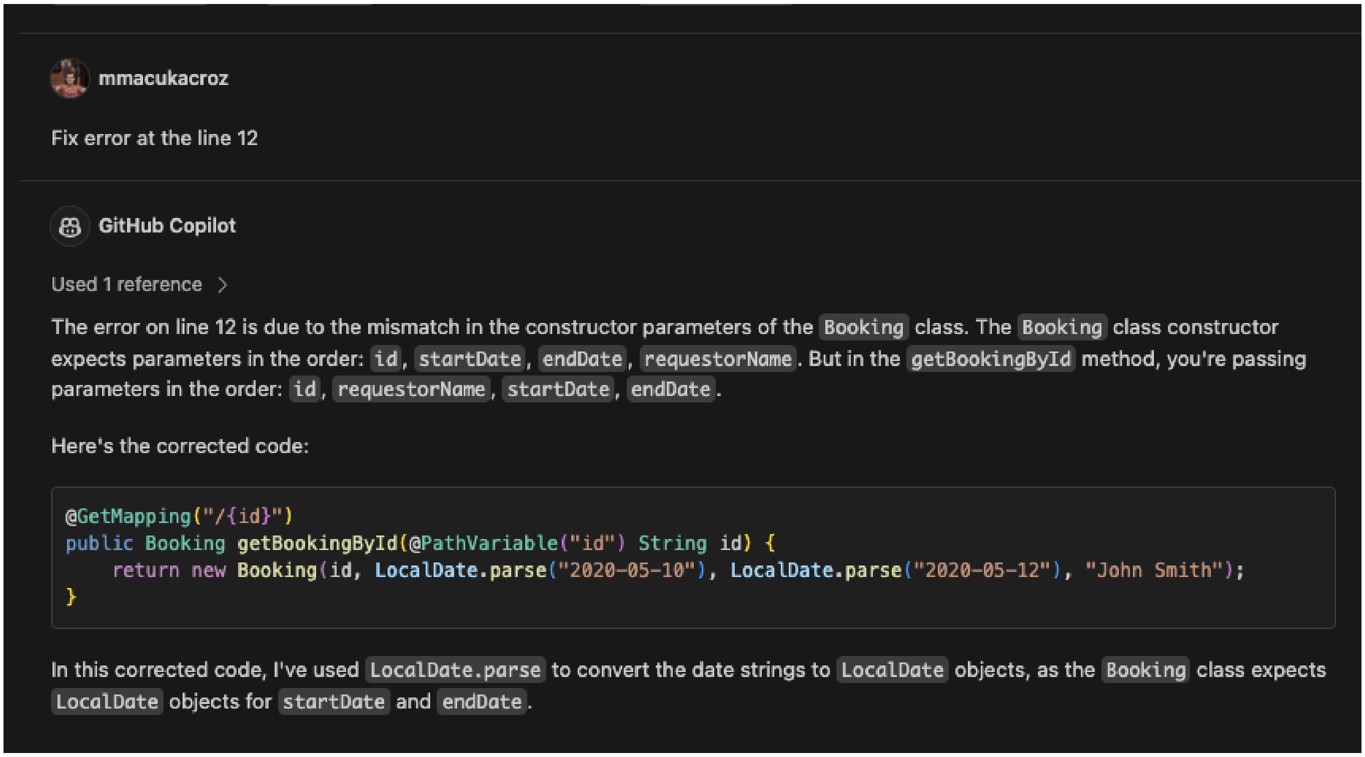


```
1   package mihael.macuka.aidemo.controller;
2
3   import mihael.macuka.aidemo.model.Booking;
4   import org.springframework.web.bind.annotation.GetMapping;
5   import org.springframework.web.bind.annotation.PathVariable;
6   import org.springframework.web.bind.annotation.RestController;
7
8   @RestController("/bookings")
9   public class DemoController {
10      @GetMapping("/{id}")
11      public Booking getBookingById(@PathVariable("id") String id) {
```

/fix The constructor Booking(String, String, String, String, String) is undefined

To solve the problem, I would move the `Booking` class to a separate file and update the package declaration in the `DemoController` class to match the expected package.

```
12          return new Booking(id, "Hotel California", "2020-05-10", "2020-05-12", "John Smith");
13      }
14  }
15
```



mmacukacroz

Fix error at the line 12

GitHub Copilot

The error on line 12 is due to the mismatch in the constructor parameters of the `Booking` class. The `Booking` class constructor expects parameters in the order: `id`, `startDate`, `endDate`, `requestorName`. But in the `getBookingById` method, you're passing parameters in the order: `id`, `requestorName`, `startDate`, `endDate`.

Here's the corrected code:

```
@GetMapping("/{id}")
public Booking getBookingById(@PathVariable("id") String id) {
    return new Booking(id, LocalDate.parse("2020-05-10"), LocalDate.parse("2020-05-12"), "John Smith");
}
```

In this corrected code, I've used `LocalDate.parse` to convert the date strings to `LocalDate` objects, as the `Booking` class expects `LocalDate` objects for `startDate` and `endDate`.