**API-Java Development Procedure**

**Step 1:** Download and Install Java JDK (Java Runtime Environment):

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

*Note: Eclipse needs Java JDK*

**Step 2:** Add Java to your path (Environment Variables): Go to System and then Advanced System Settings and click on Environment variables under the Advanced Tab.

Finally, in the Environment Variables window, highlight the path  
variable in the Systems Variable section and click edit. Add the  
directory where the java executable is: **C:\Program Files\Java\jdk-11\bin**

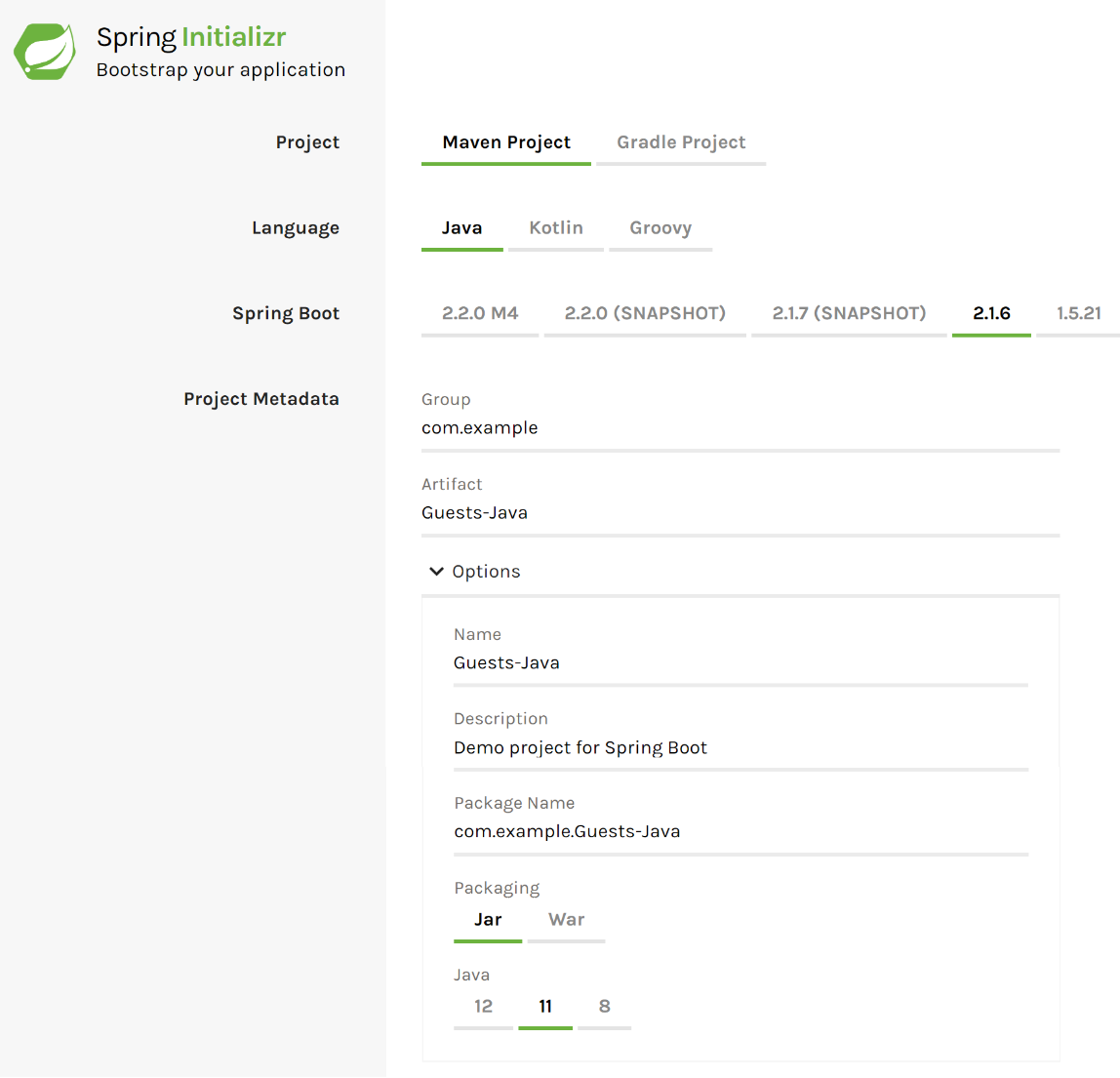
**Step 3:** Open Command Line and Check the Java Version: C:\Users\ahmed>**java -version**

*Note: Command Line Previous Folder Comman*d*:* ***CD ..***

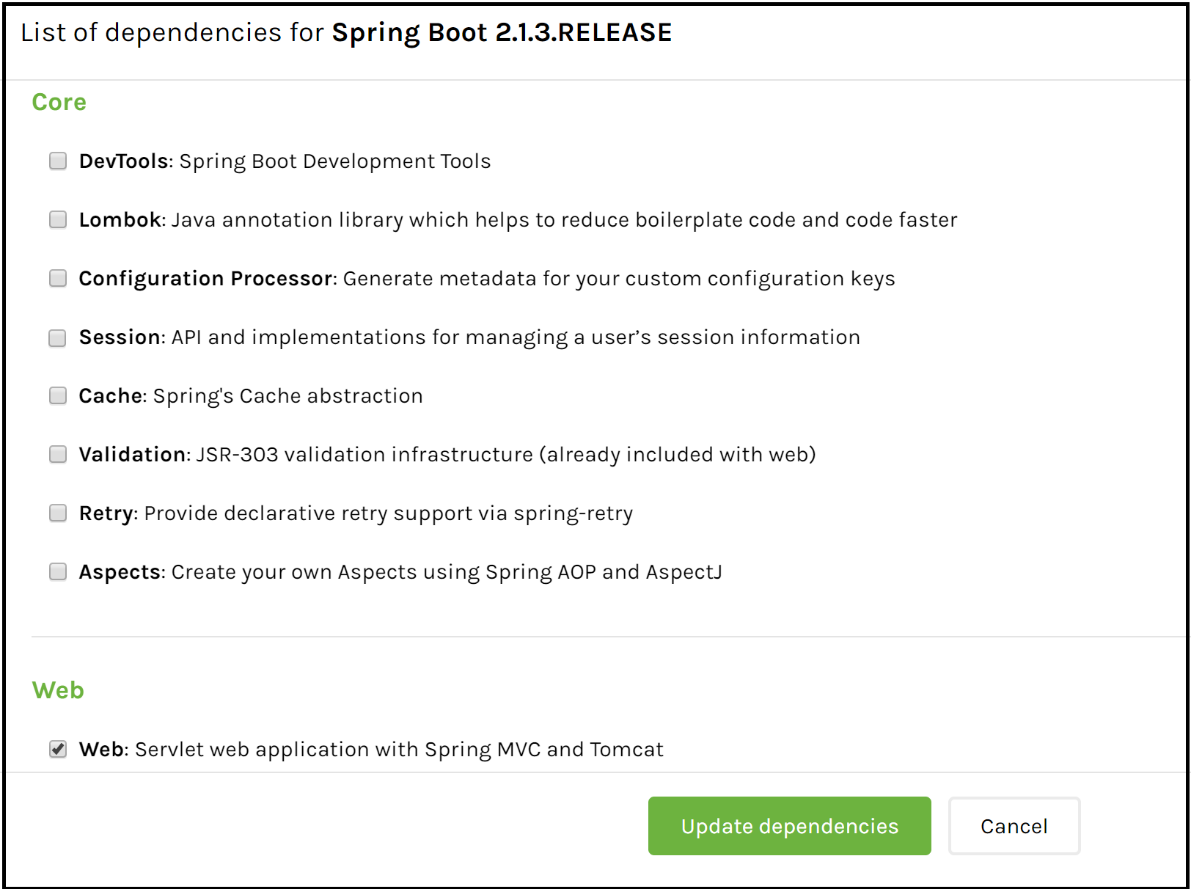
*Note: Command Line First Folder Comman*d: ***CD \***

**Step 4: Install IntelliJ IDE or Eclipse IDE.**

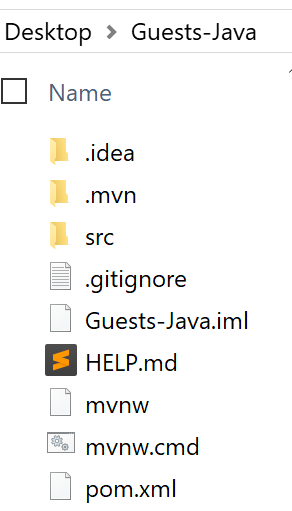
**Step 5:** Go to<https://start.spring.io/> to initialize Java’s **Spring Boot** framework!



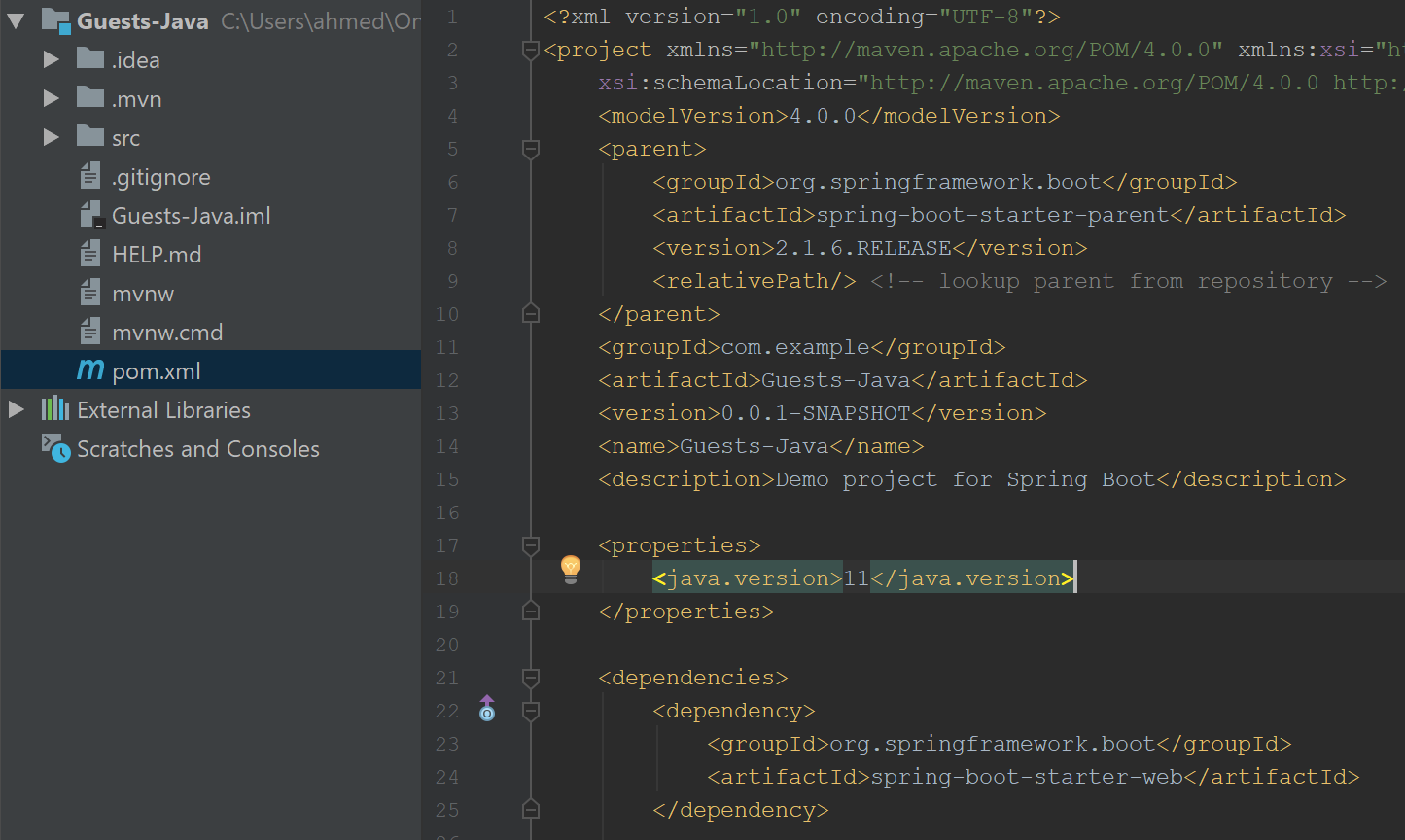
**Step 6:** Choose your dependencies before **“Generating Project”** for rapid development. Settings for dependencies are set up in the Generated Project for you. Saving you time.



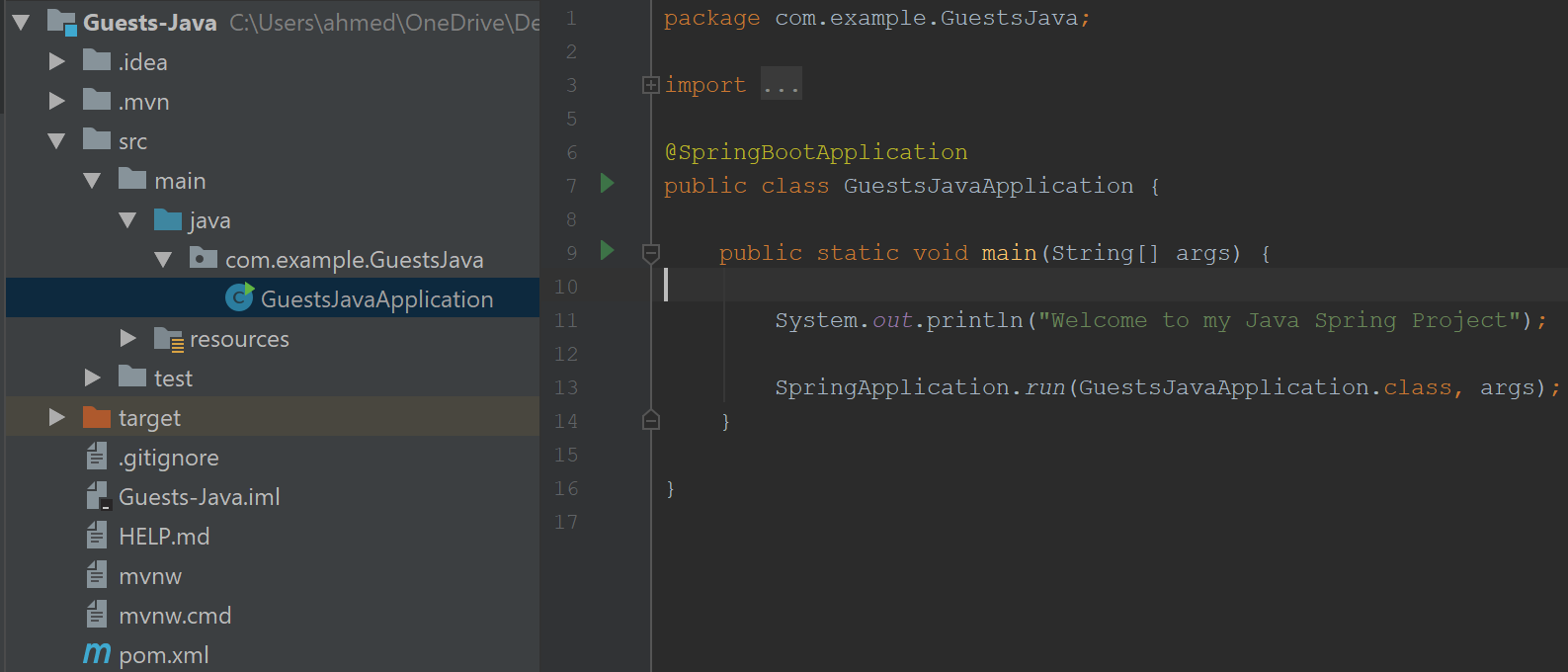
**Note\*** Here is the folder structure of the Spring framework.



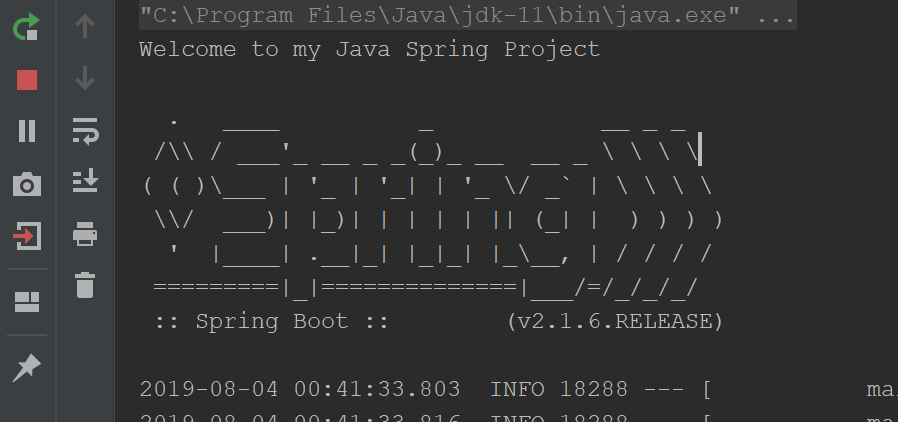
**Note\*** Here is the pom.xml file. (Project Object Model) **(Similar to package.json in JS)**



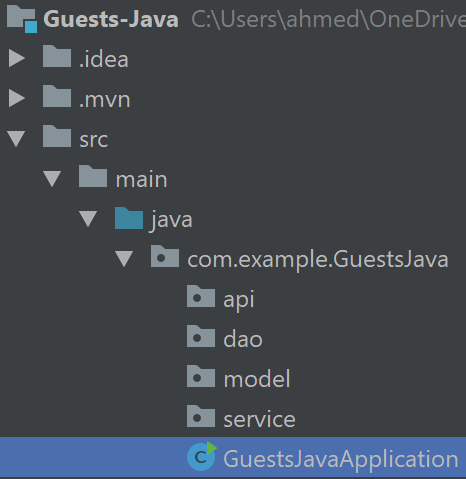
**Step 7:** Add the following code and Run your Java Application: (Running **GuestsJavaApplication**)



**Server View:**

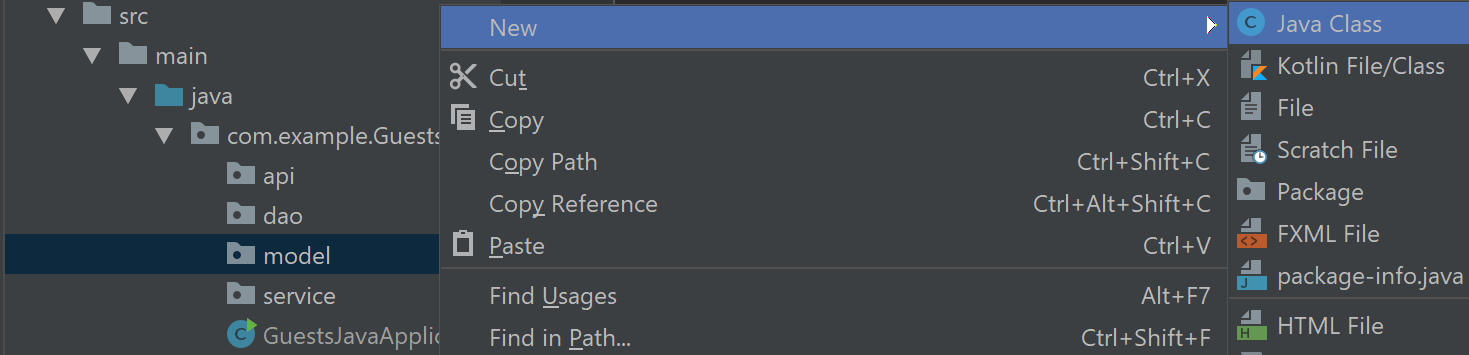


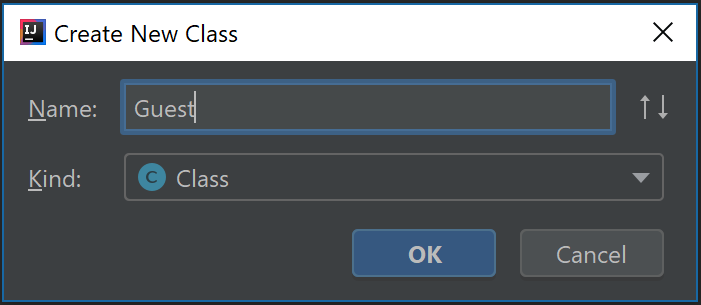
**Step 8:** Create 4 **packages** called **api**, **dao**, **model** and **service** in the following location:



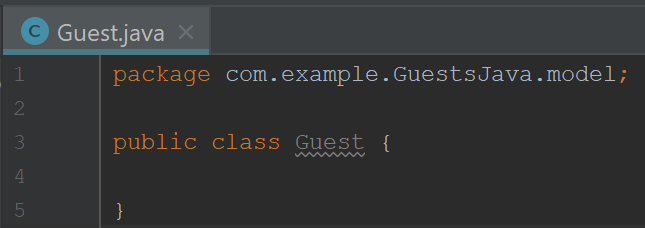
**Creating the Model**

**Step 9:** Right click on the model package and create a new Java Class.

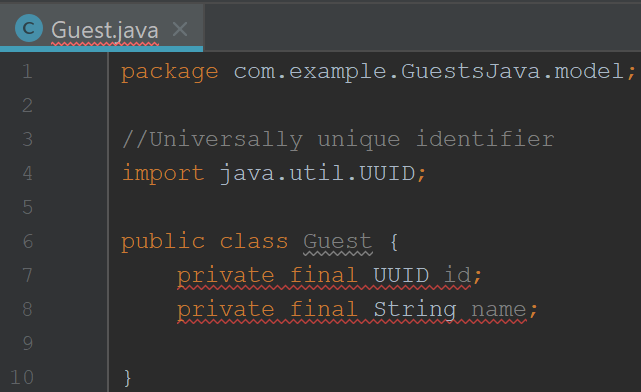




**Java Class File**:

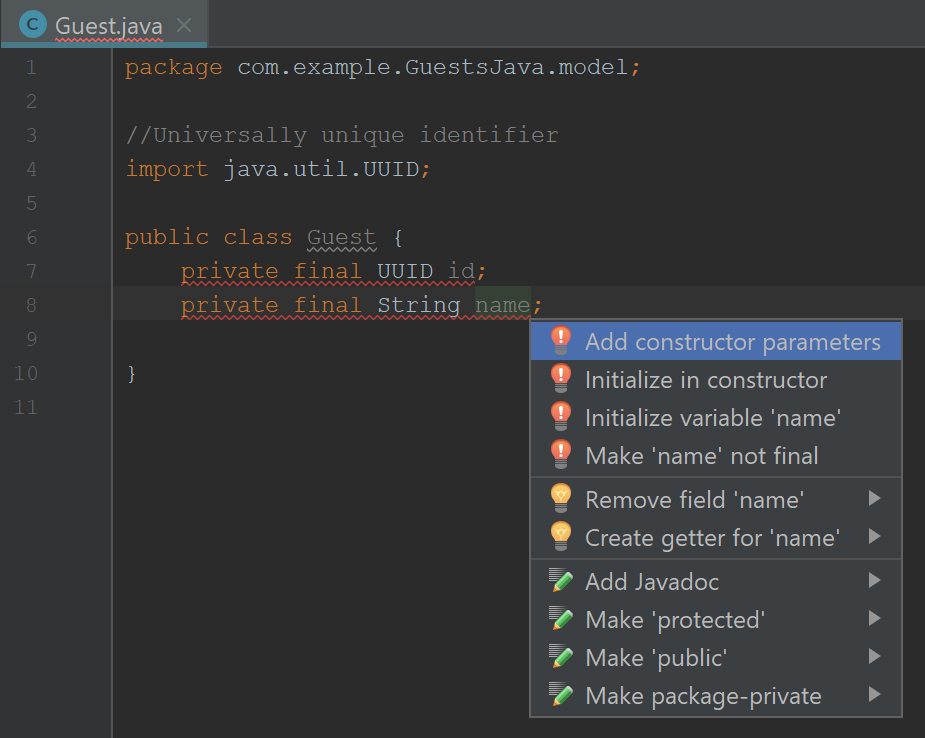


**Step 10:** Add 2 Private Variables called **id** and **name**

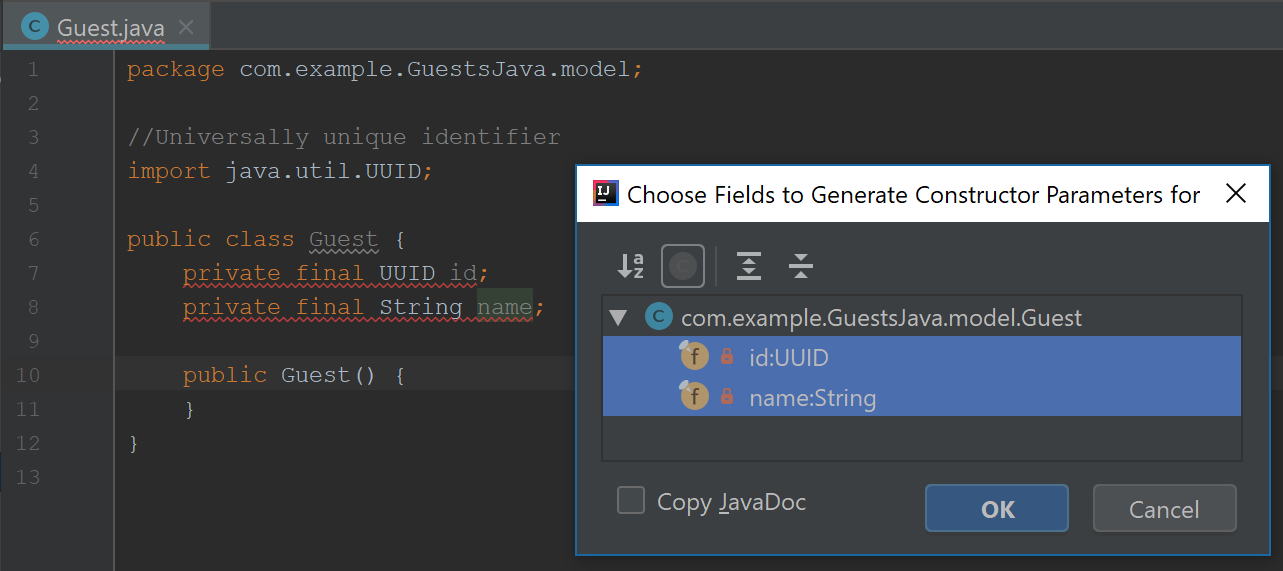


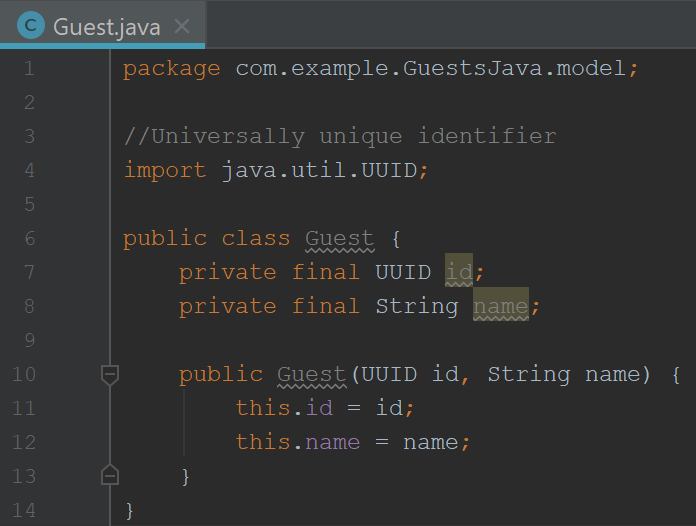
**Automatically Added.**

**Step 11:** Press **Alt + Enter** to use the create constructor shortcut.



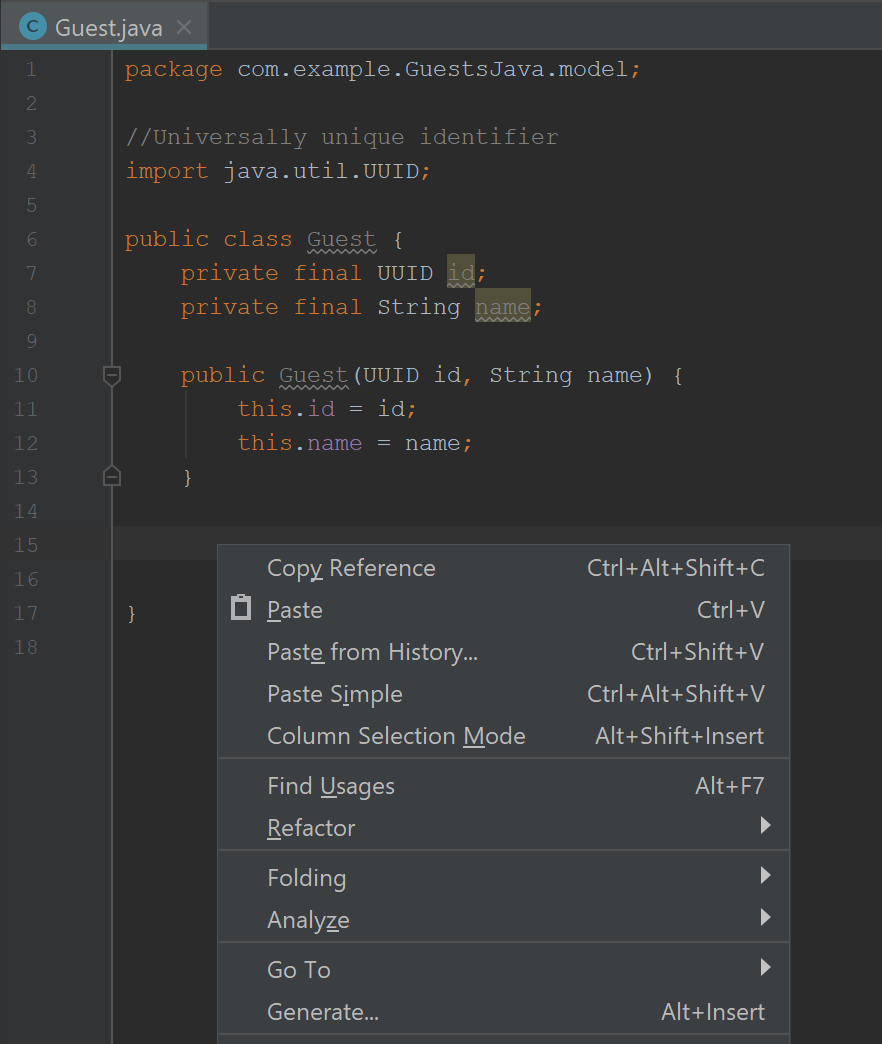
**Step 12:** Choose both fields to create constructors for.



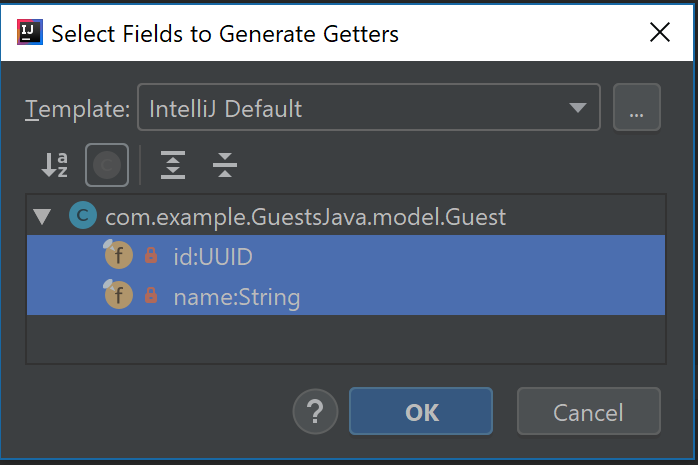


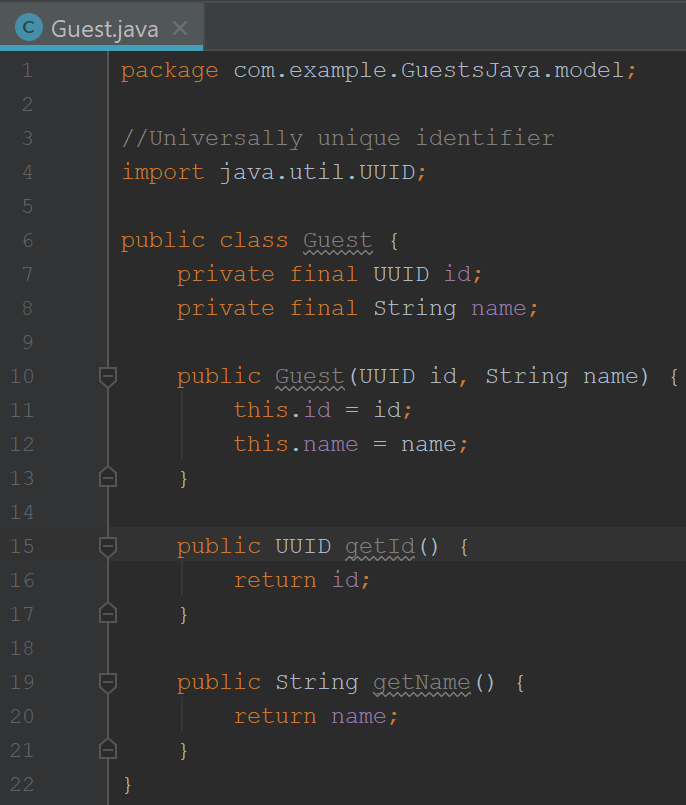
**Generated.**

**Step 13: Generate** the **Getters** by right clicking and choosing “**Generate**” (or **Alt + Insert**)



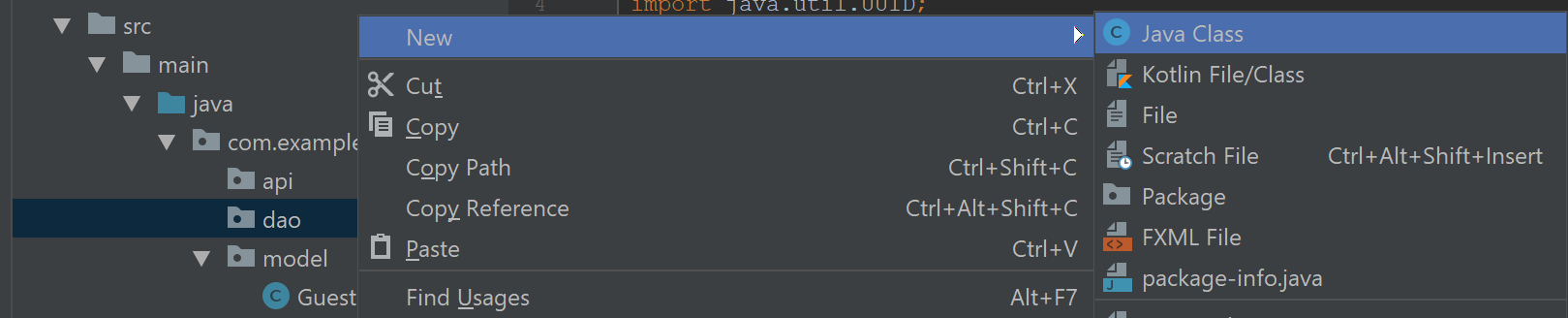
**Step 14:** Choose both fields to create getters for:

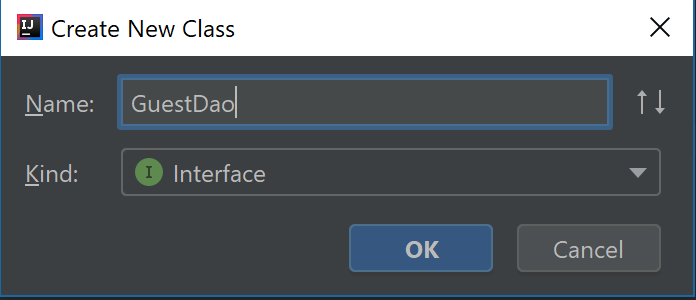


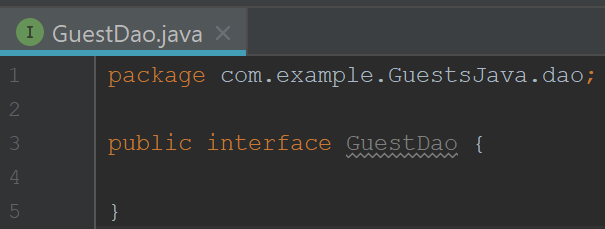


**Generated.**

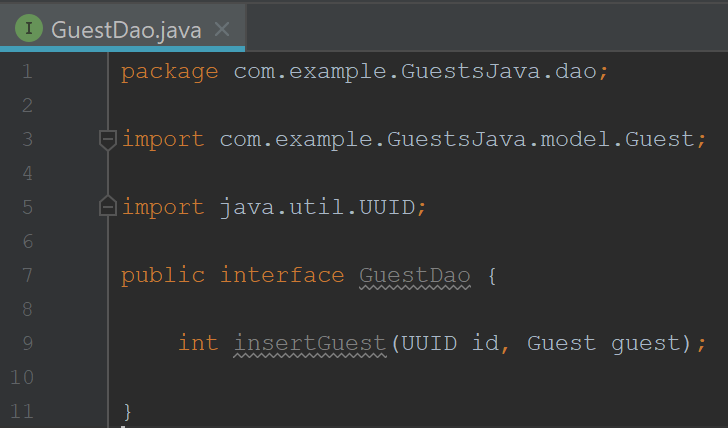
**Step 15:** Createa new Java Class called **GuestDao** of Kind **Interface** (*Instead of Class*)







**Step 16:** Create a Method to add a Guest **with an id**.

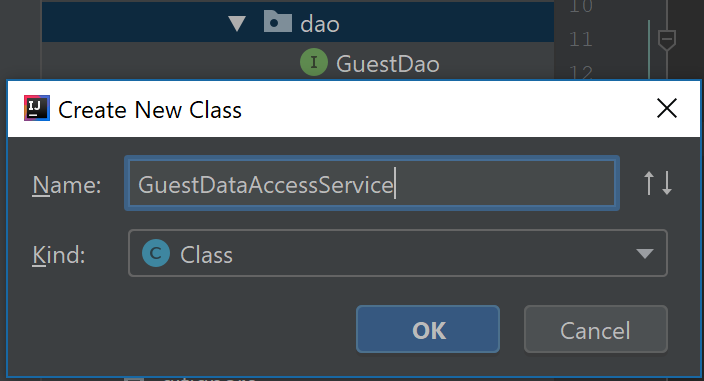


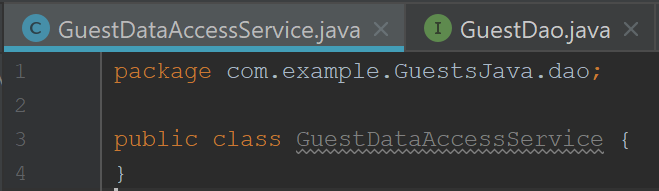
**Automatically Added.**

**Step 17:** Create a Method to add a Guest **without an id** and assign a random Id to that Guest.

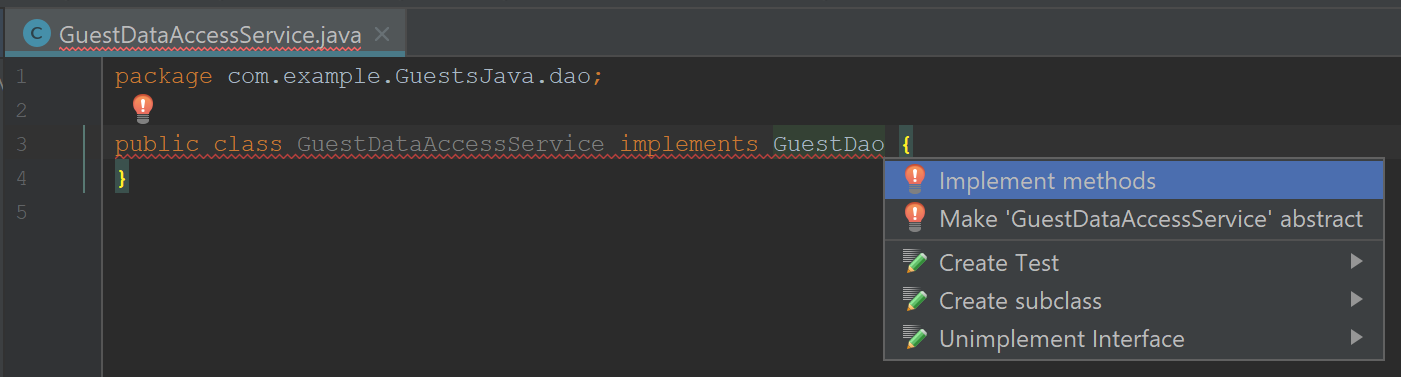


**Step 18:** Create a new class under **dao** and call it “**GuestDataAccessService**.”

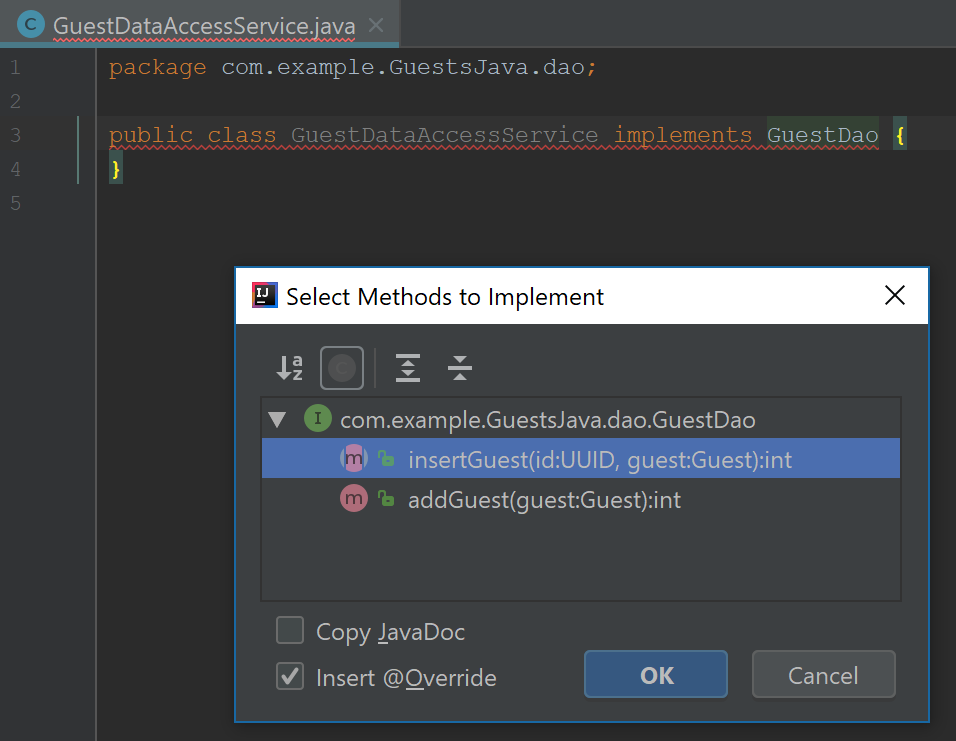




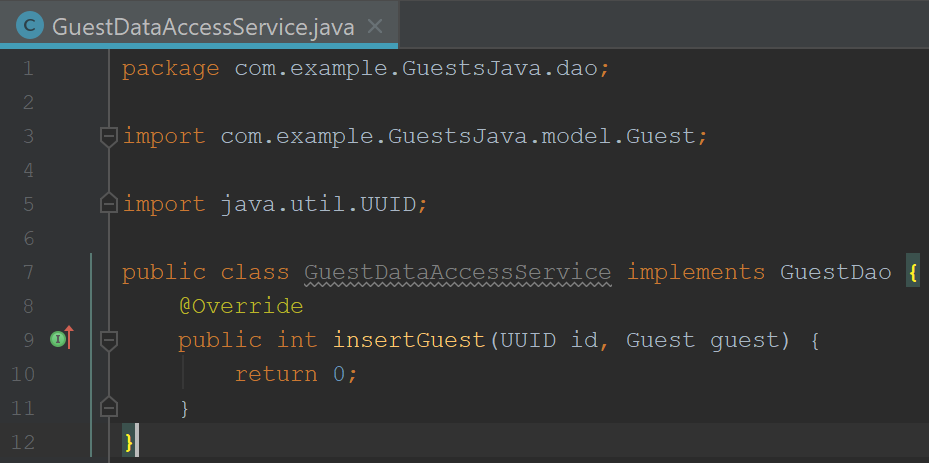
**Step 19:** Add “implements GuestDao” and press **Alt + Enter** and then click **implements method**.



**Step 20:** Implement the “**insertGuest**” method.

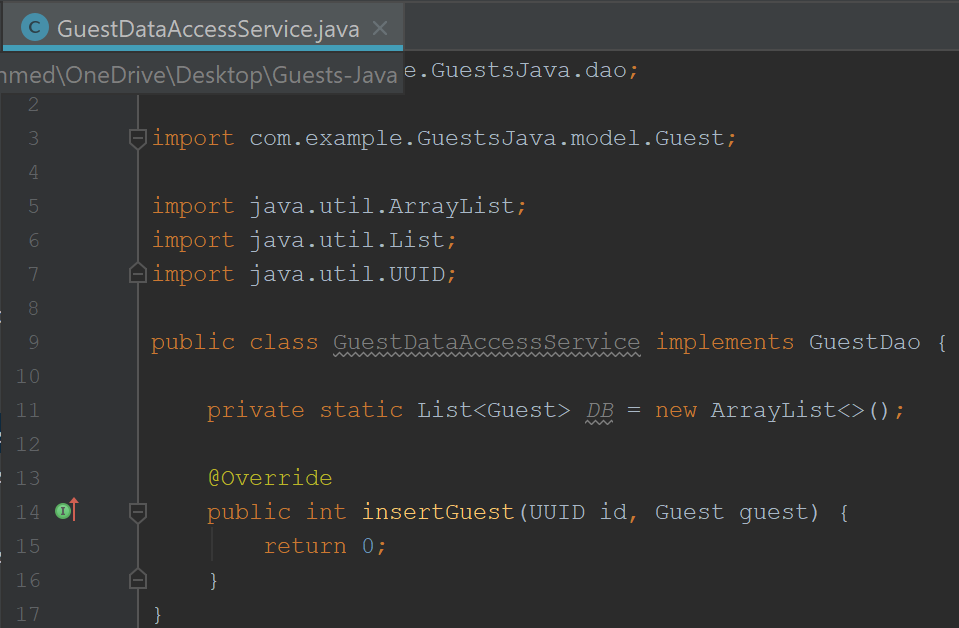


**Automatically Added.**



**Generated.**

**Step 21:** Create a guest <List> and <ArrayList>



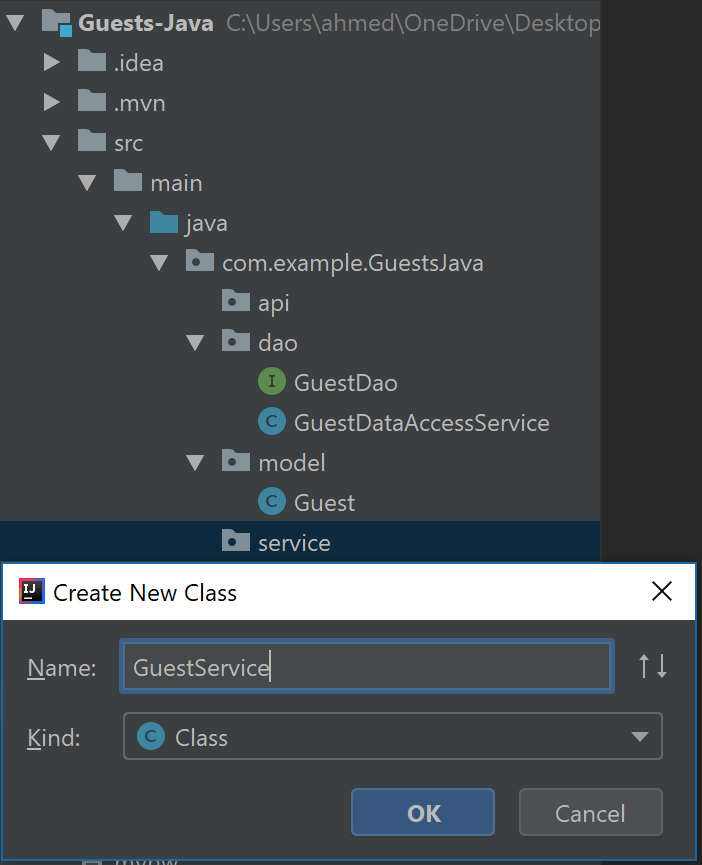
**Automatically Added.**

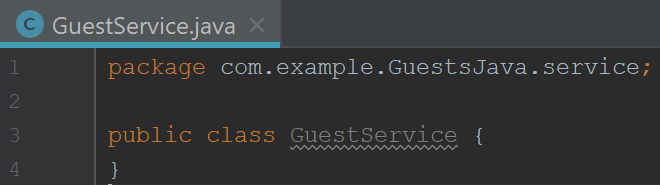
**Step 22:** Adding a Guest to the Database.

(Change Return to 1 to make sure that the insertion always works.)

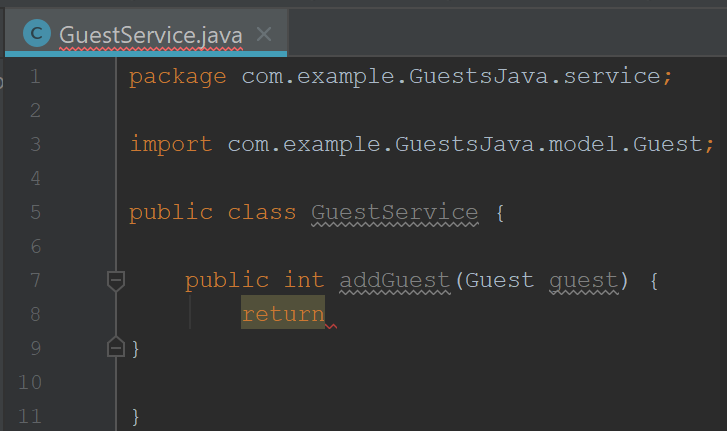


**Step 23:** Define the Service. Create a new **Class** in the **Service** Package called **GuestService.**



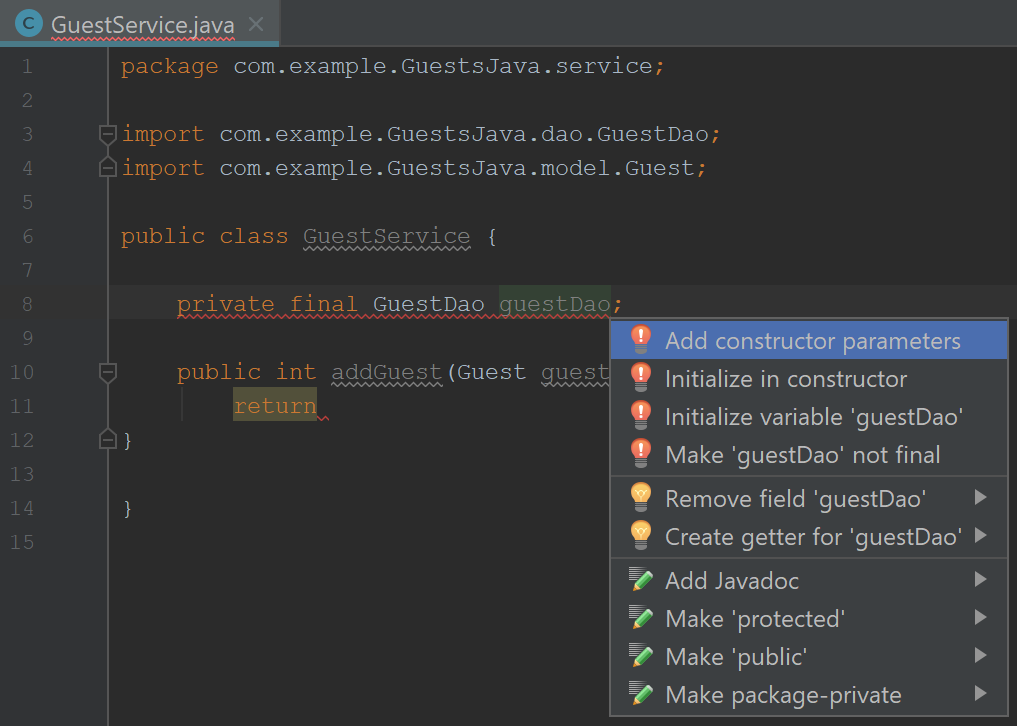


**Step 24:** WriteMethod to insert a new Guest



**Automatically Added.**

**Step 25:** Add GuestDao and Add the Constructors.



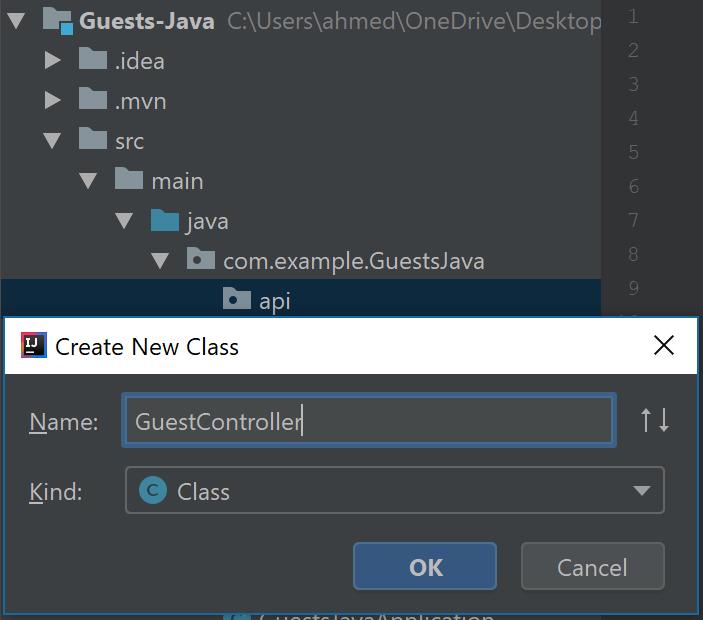
**Step 26:** Add Code to return a Guest.

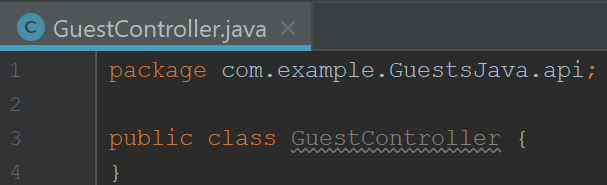


**Add Code:**

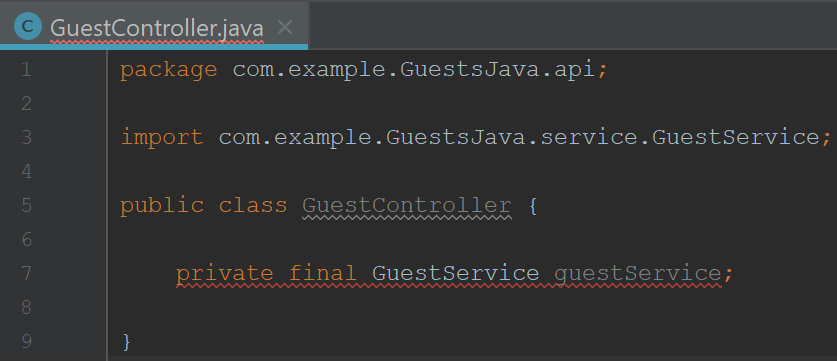
**Constructor Generated.**

**Step 27:** Creating the API. Create a **Class** in the **api** package called “**GuestController**.”

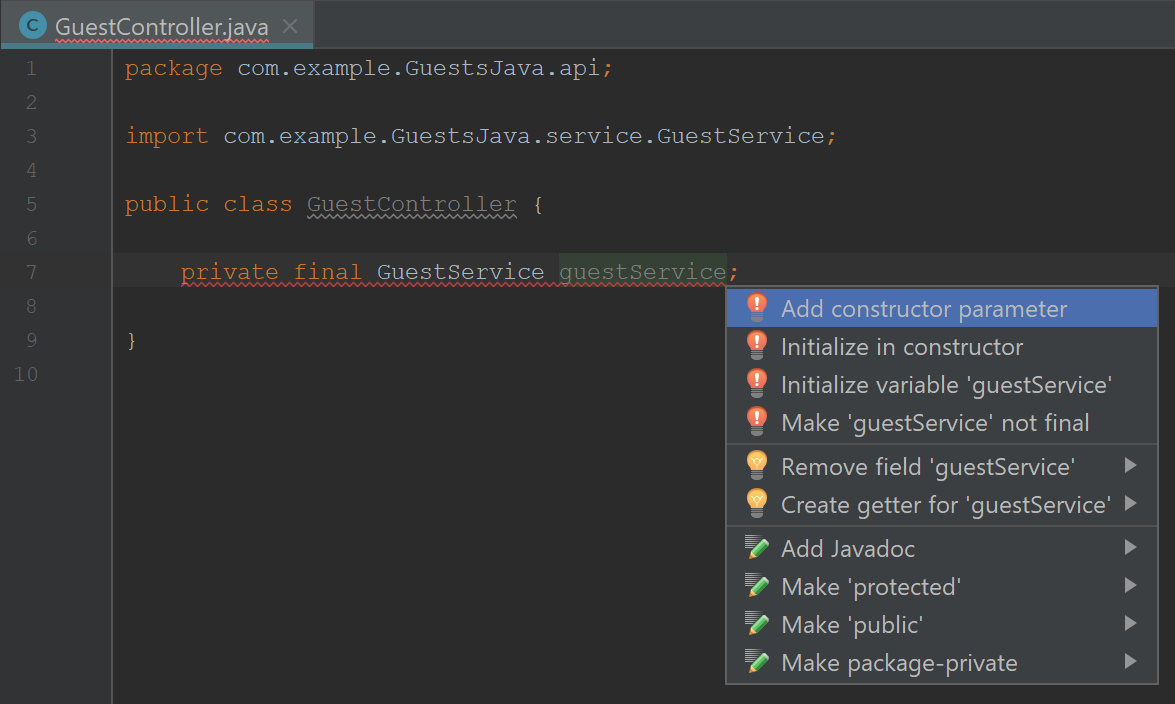


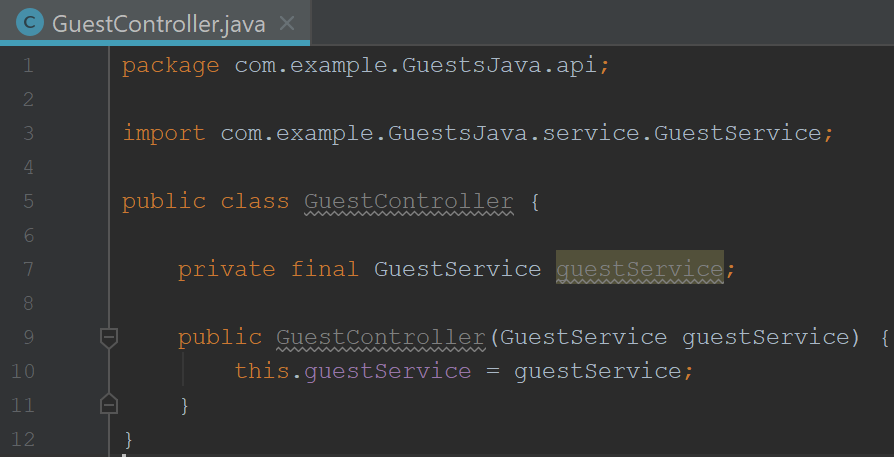


**Step 28:** Add **GuestService** to **GuestController**



**Step 29:** Add Constructor.



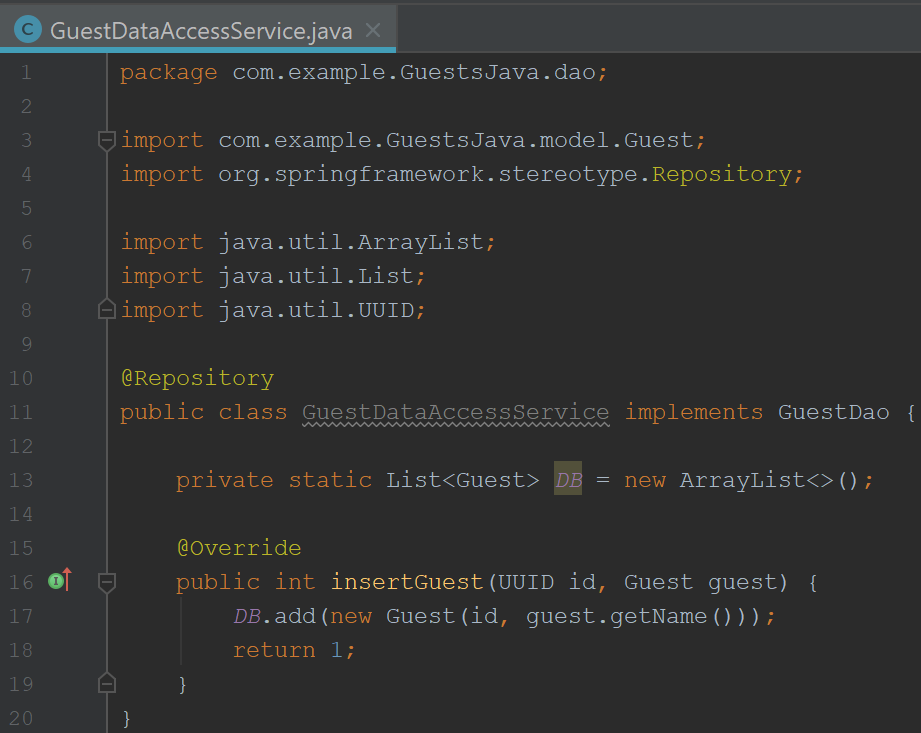


**Constructor Generated.**

**Step 30:** Add Method to add a Guest.



**Step 31:** Add **@Repository** (Class needs to be instantiated in order to be injected in other classes)



**Automatically Added.**

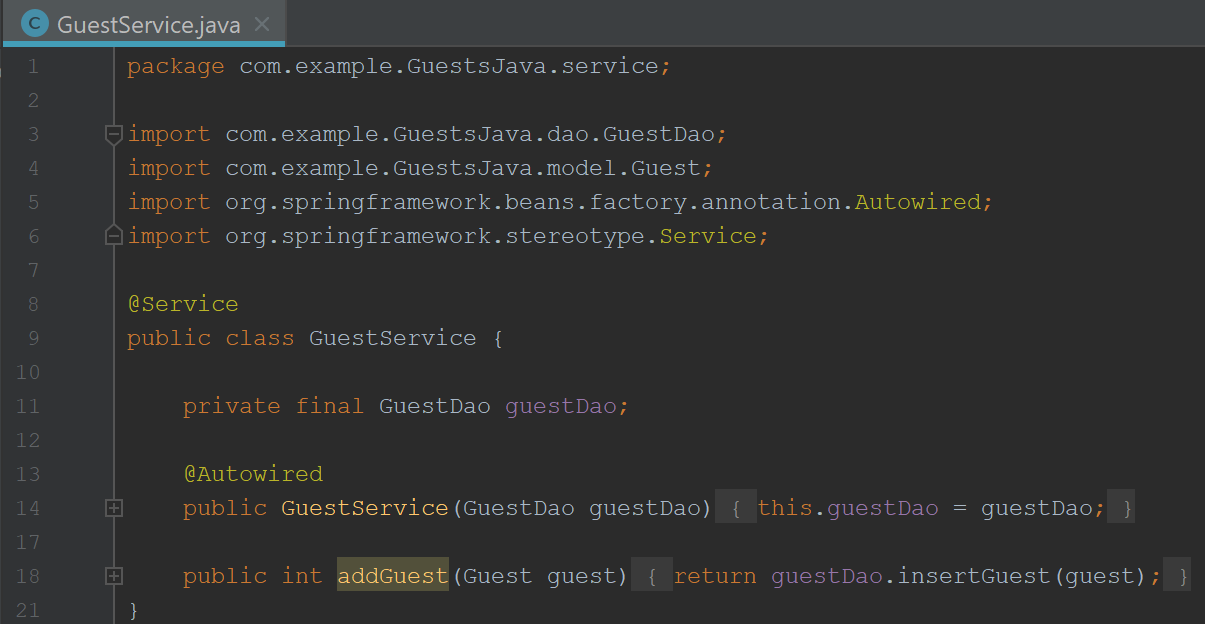
**Step 32:** Add **@Service**



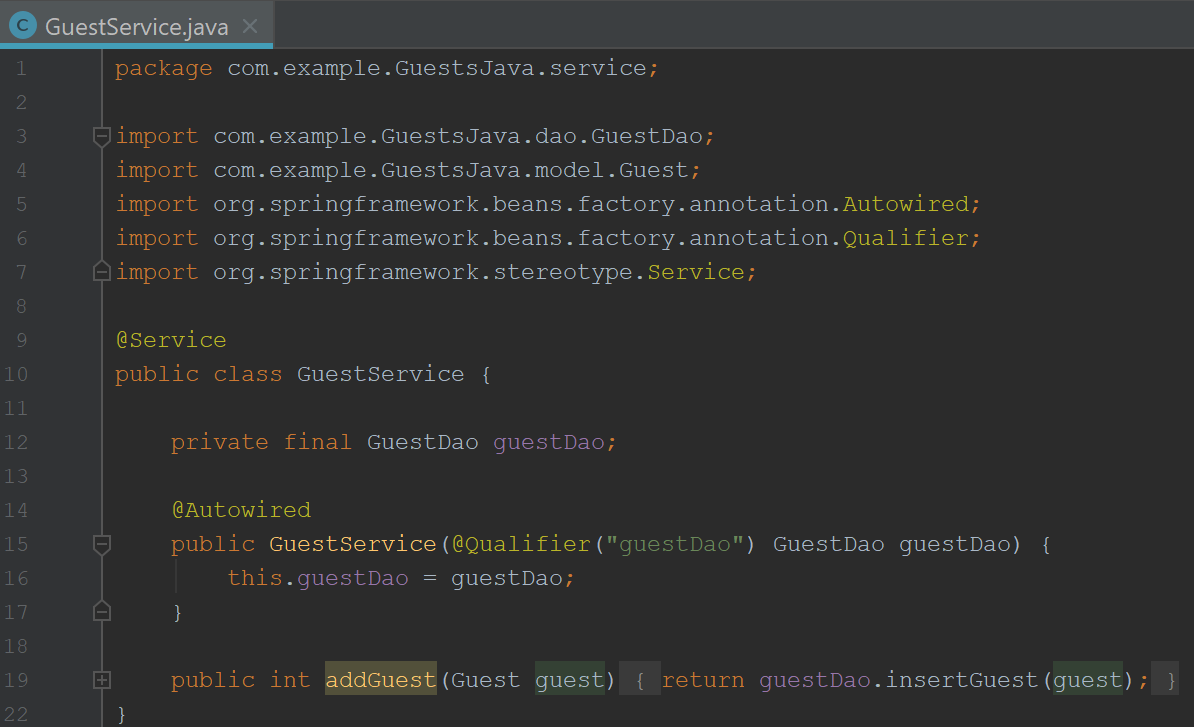
**Automatically Added.**

**Step 33:** Add **@AutoWired** in order to inject into the constructor.

**Automatically Added.**

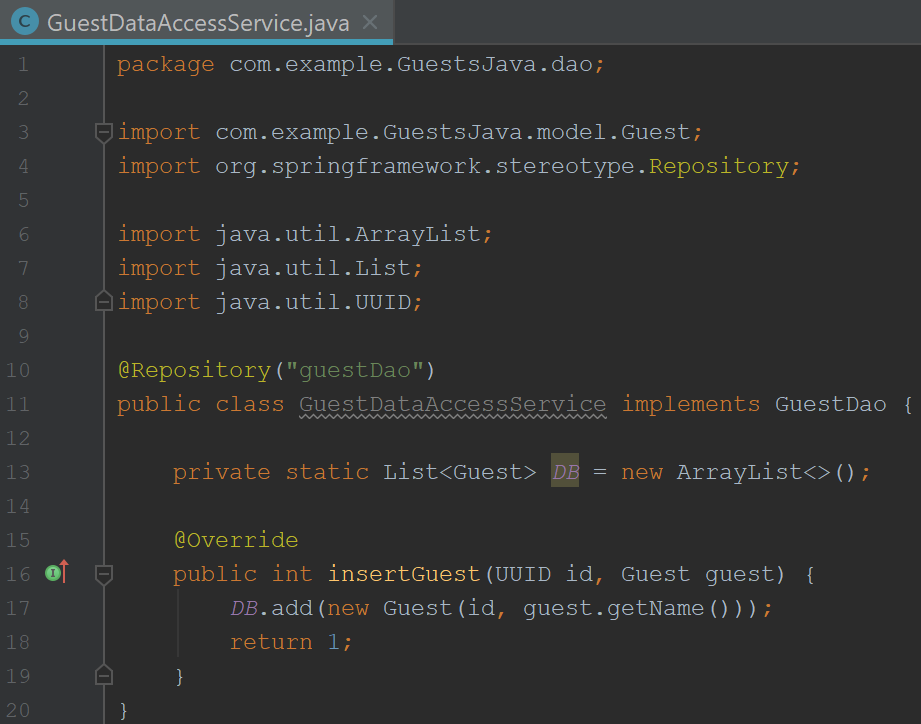


**Step 34:** Add **@Qualifier(guestDao)** to help the interface distinguish database implementations.



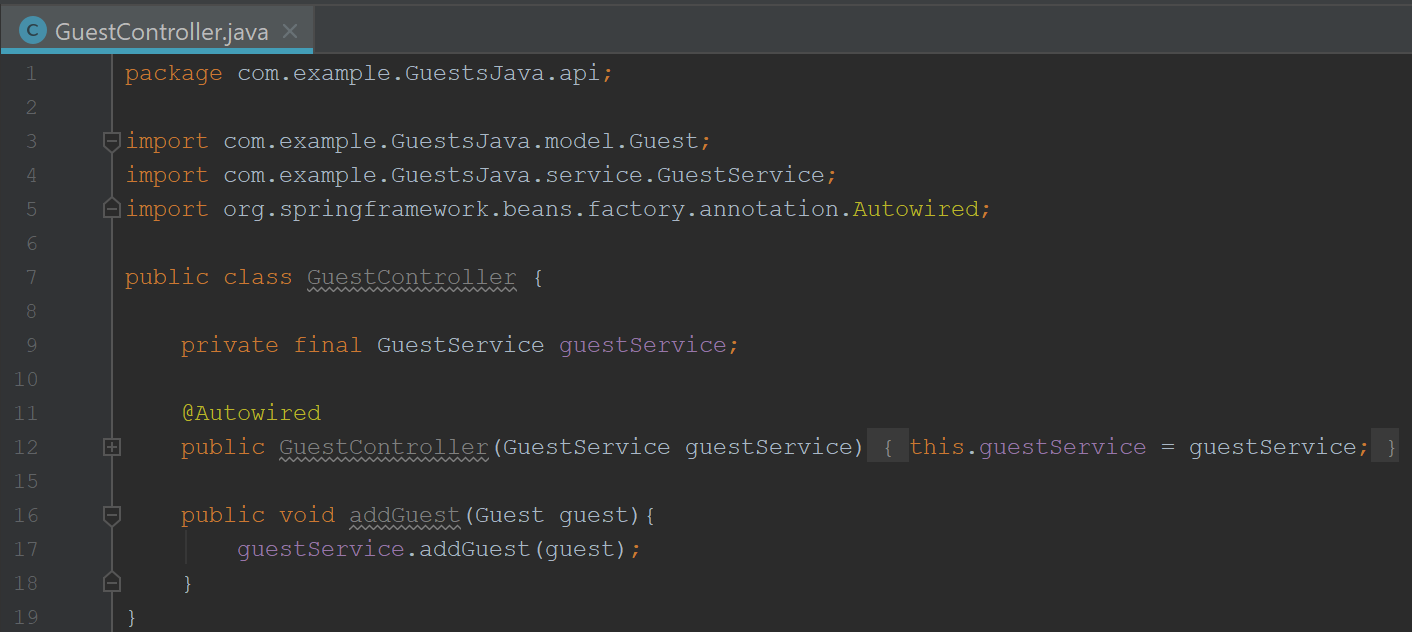
**Automatically Added.**

**Step 35:** Give the repository the same name.

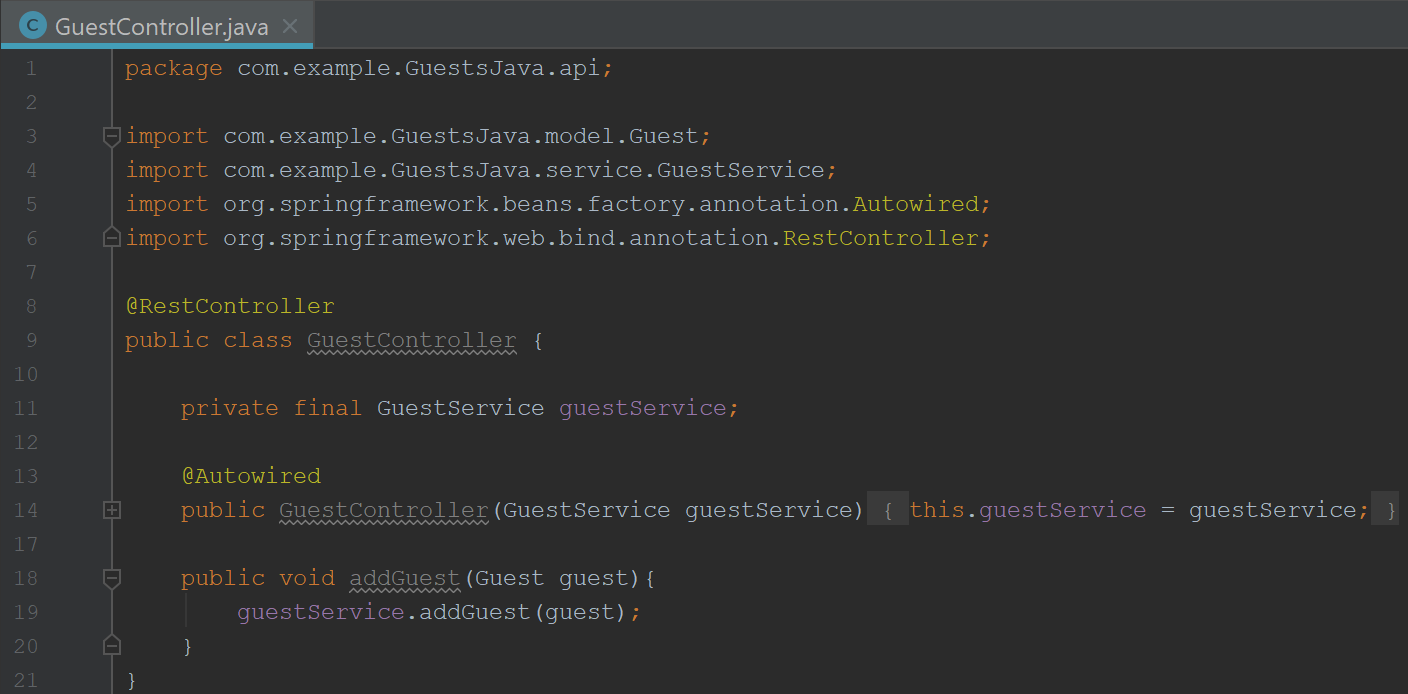


**Automatically Added.**

**Step 36:** Add **@Autowired** and Spring will inject the service into the constructor.

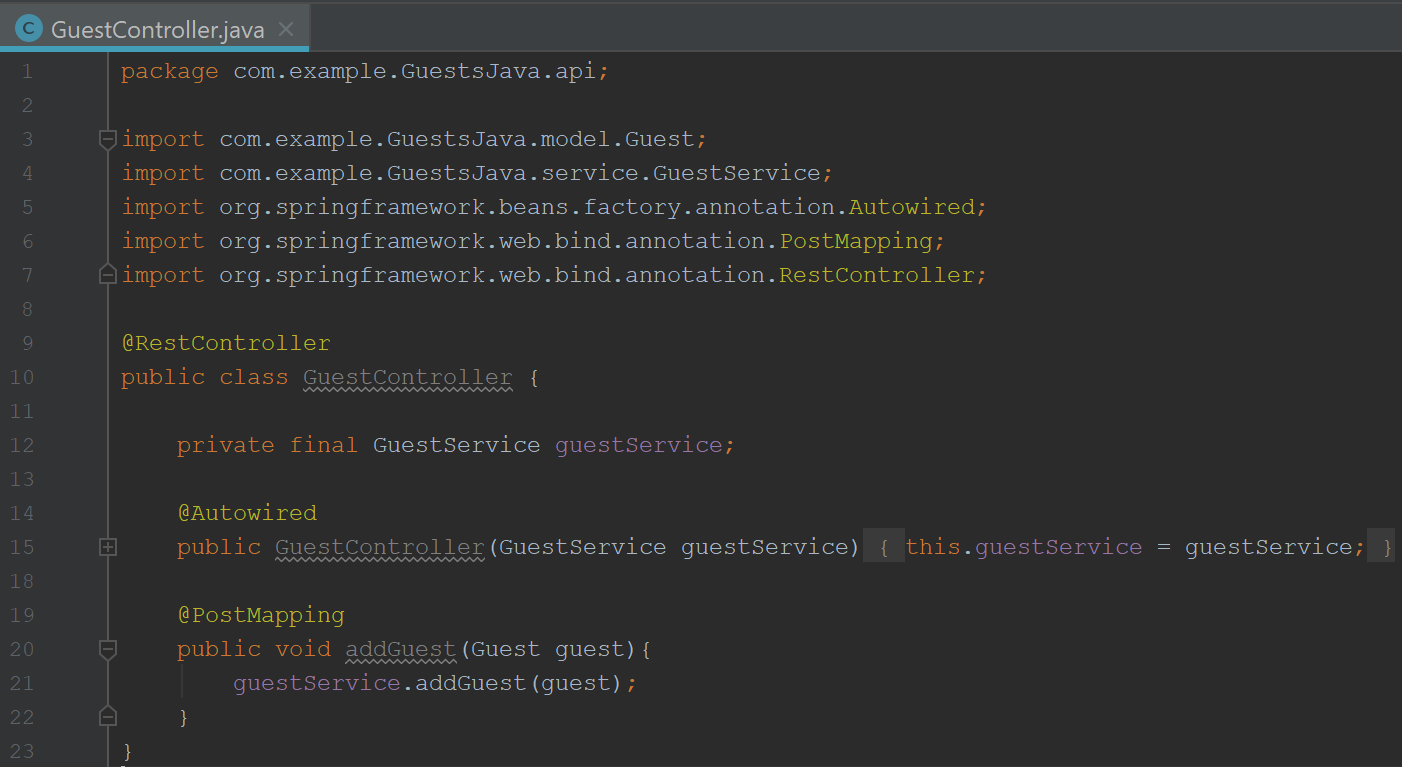


**Step 37:** Add **@RestController** to allow RESTful Web Services/HTTP Calls.

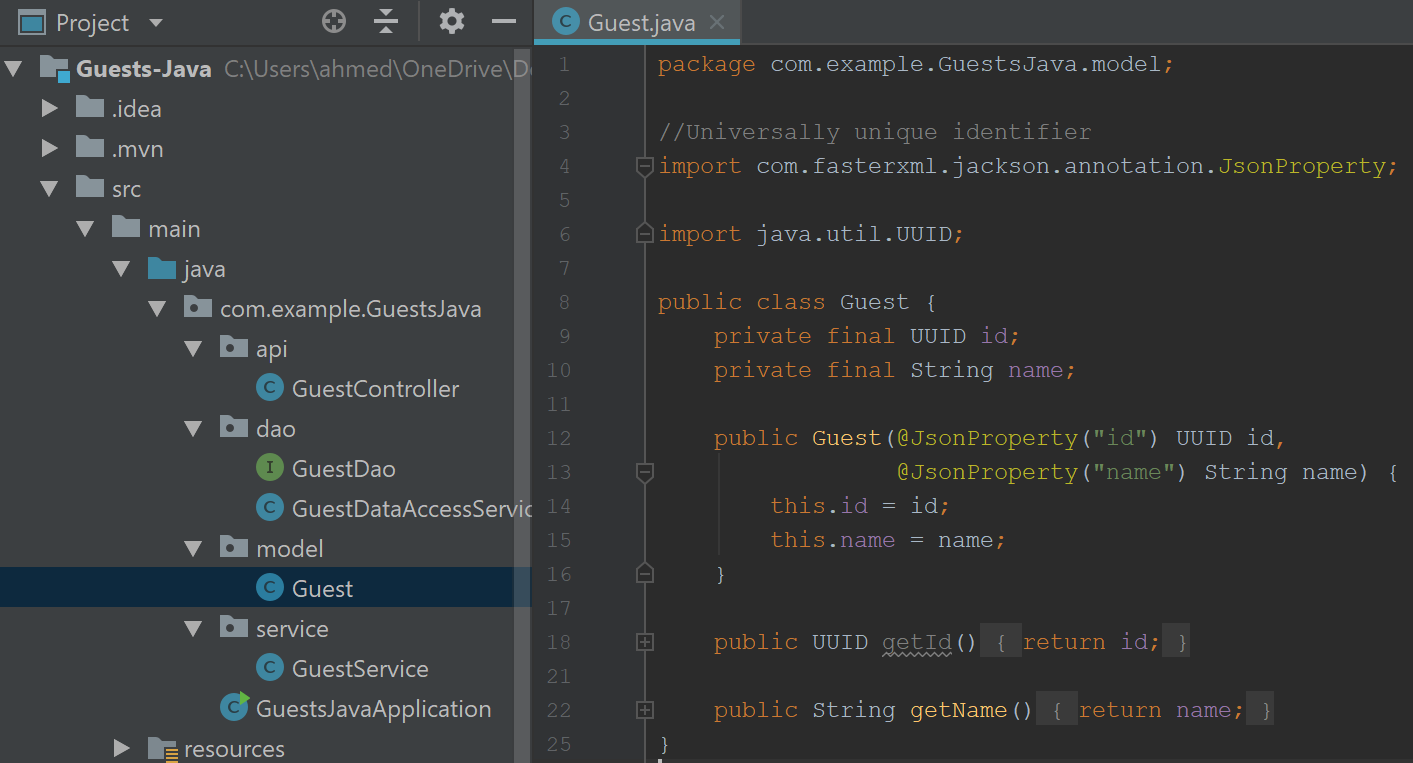


**Step 38:** Add **@PostMapping** to tell Spring that the method will be used for Posts.

**Automatically Added.**

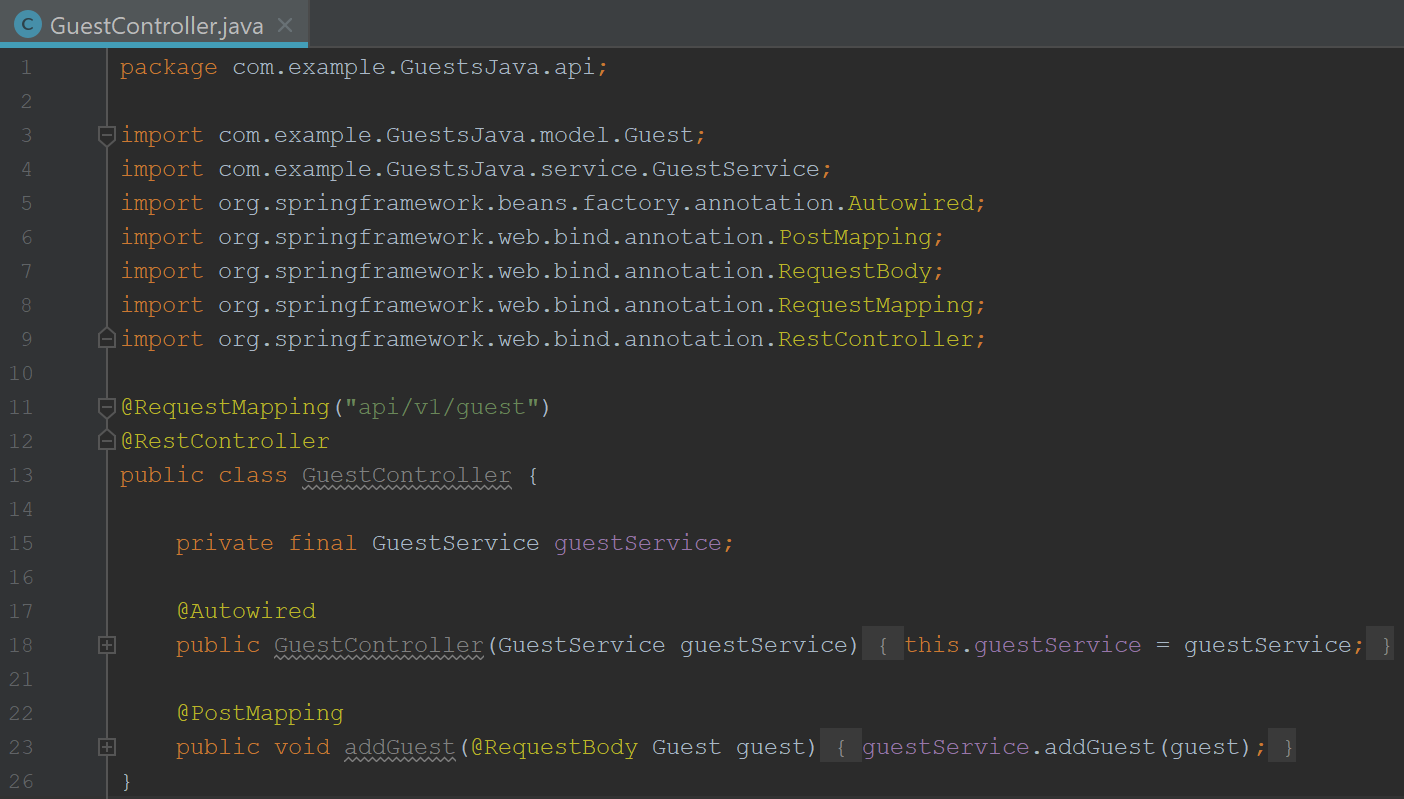


**Step 39:** Add **@JsonProperty** to signify the data property as Json.

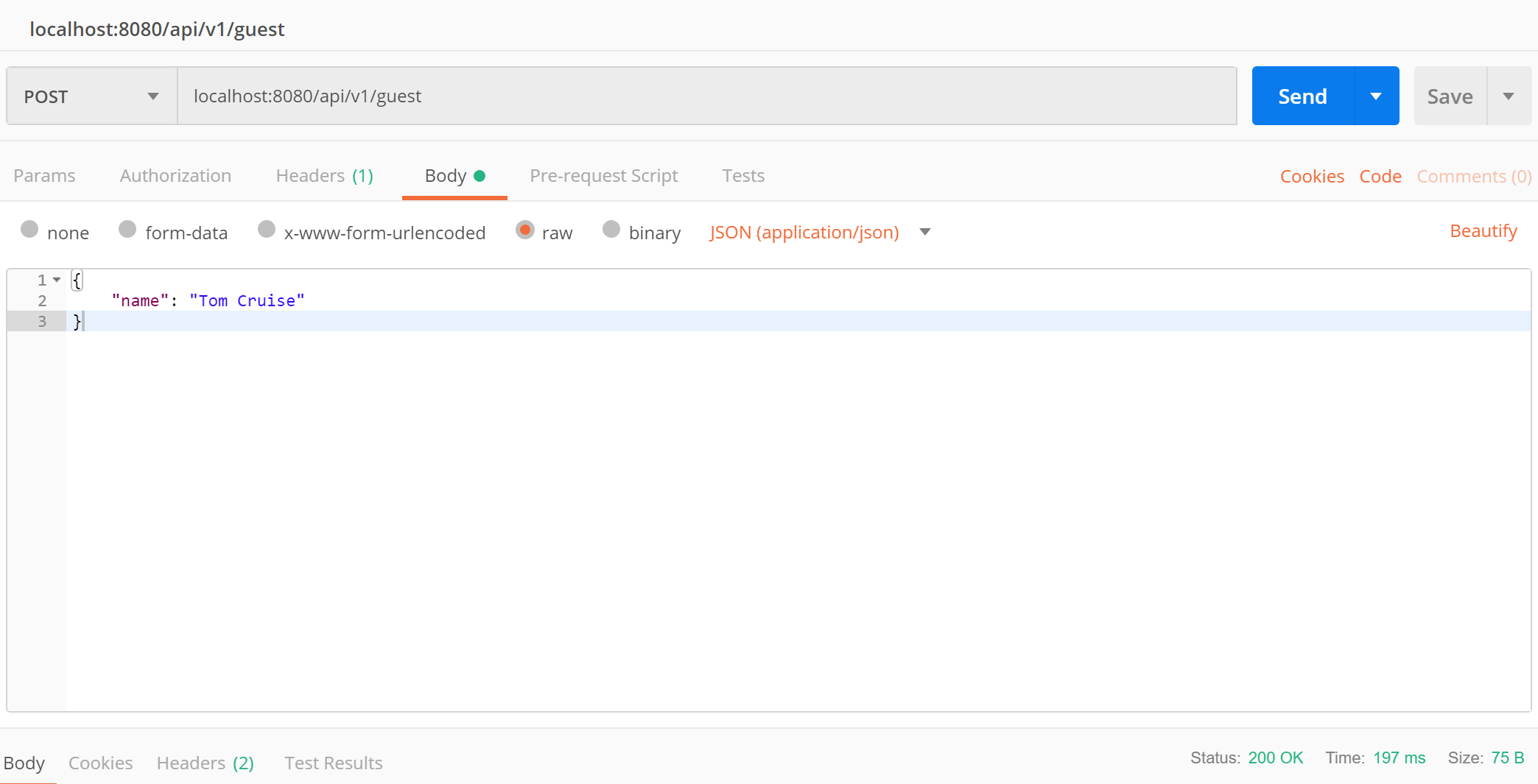


**Automatically Added.**

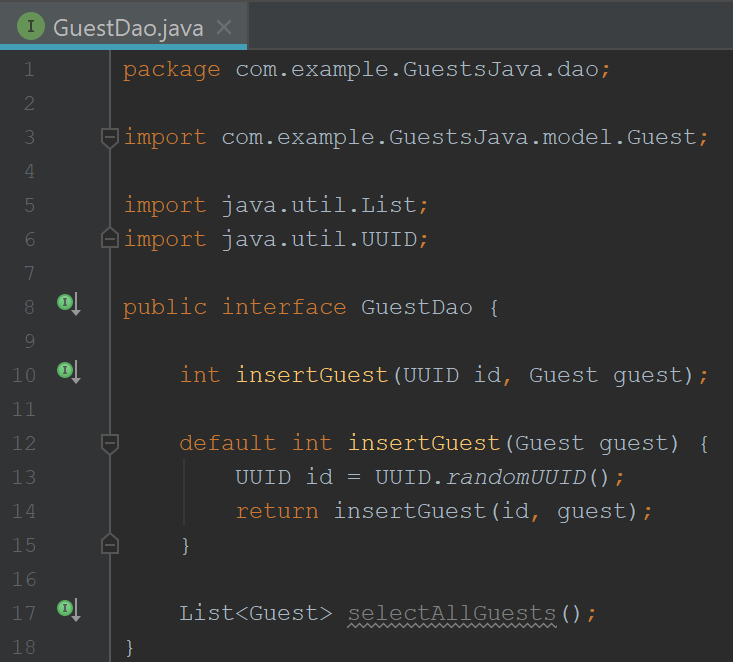
**Step 40:** Add **@RequestBody** and **@RequestMapping.**



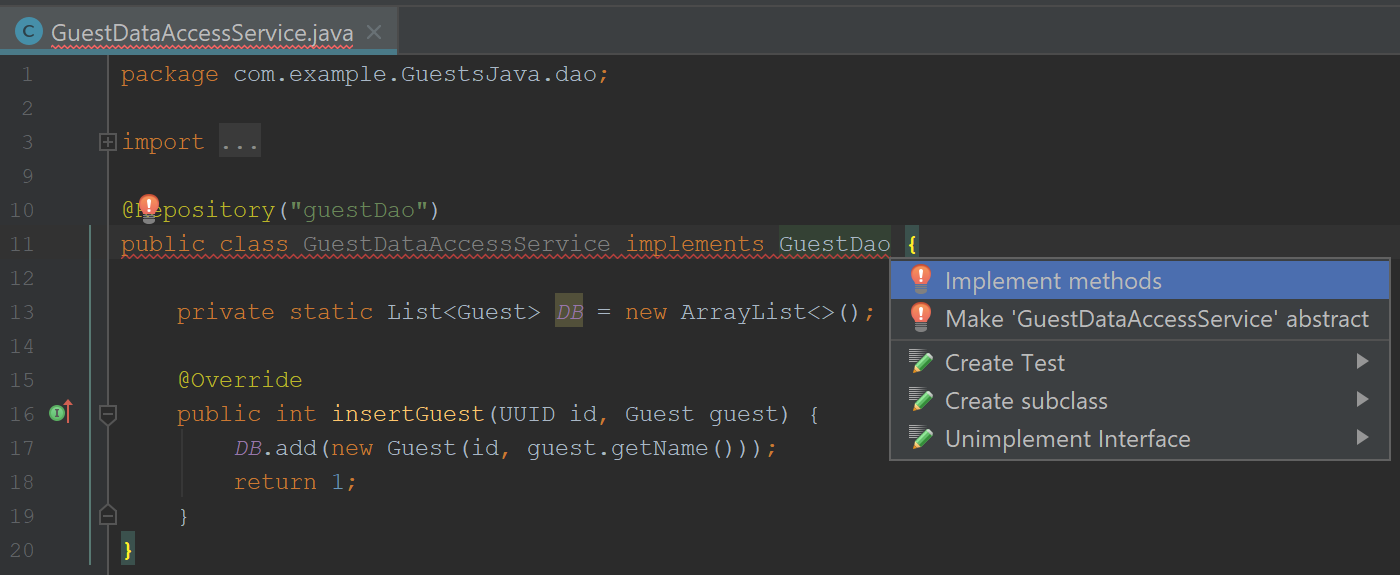
**Step 41:** Test Posting Using **Postman Software.**

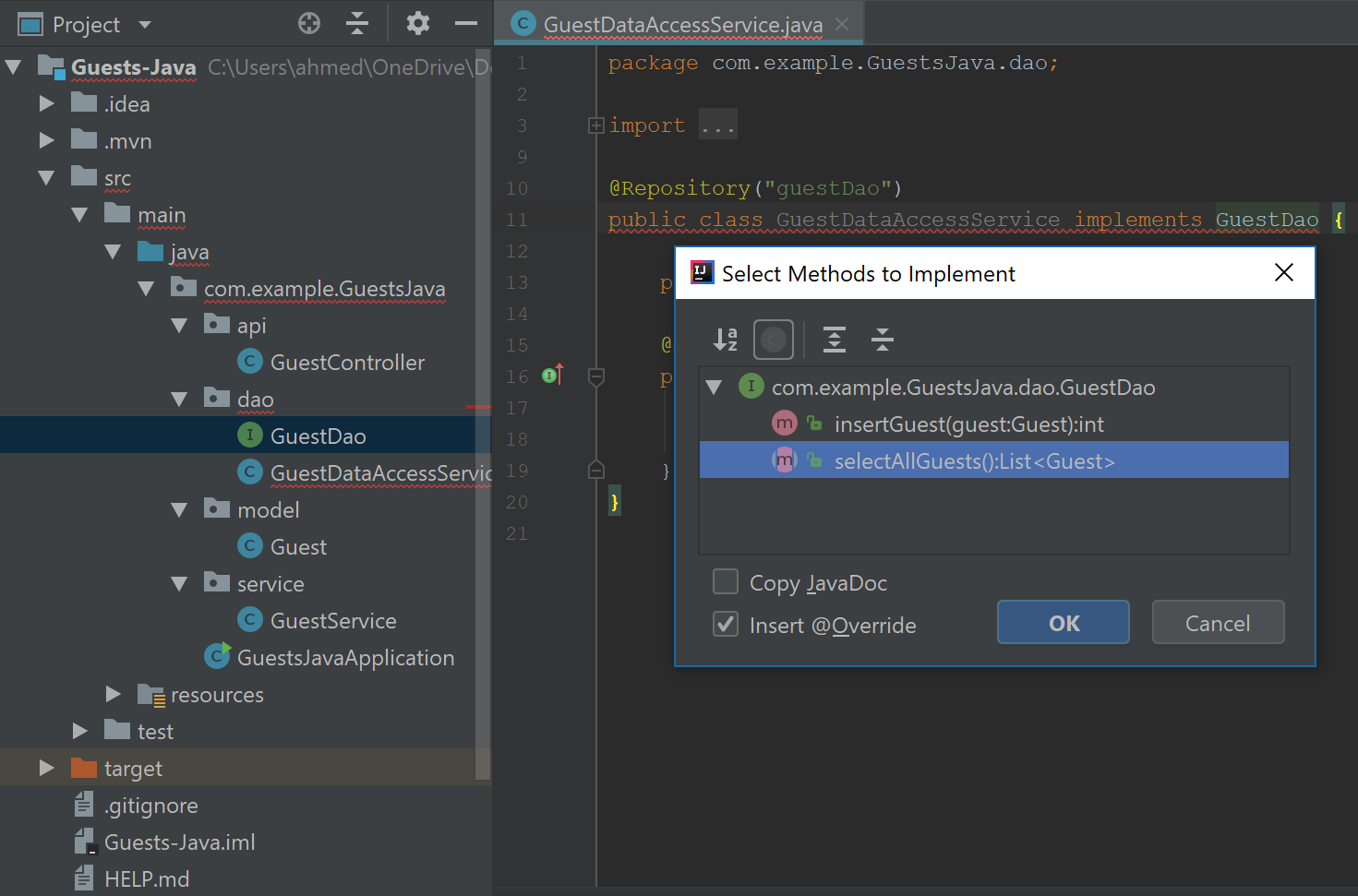


**Step 42:** Create a **List** to hold the Guests and create the **selectAllGuests()** Method.



**Step 43:** Implement the Method **selectAllGuests()** in GuestDao.

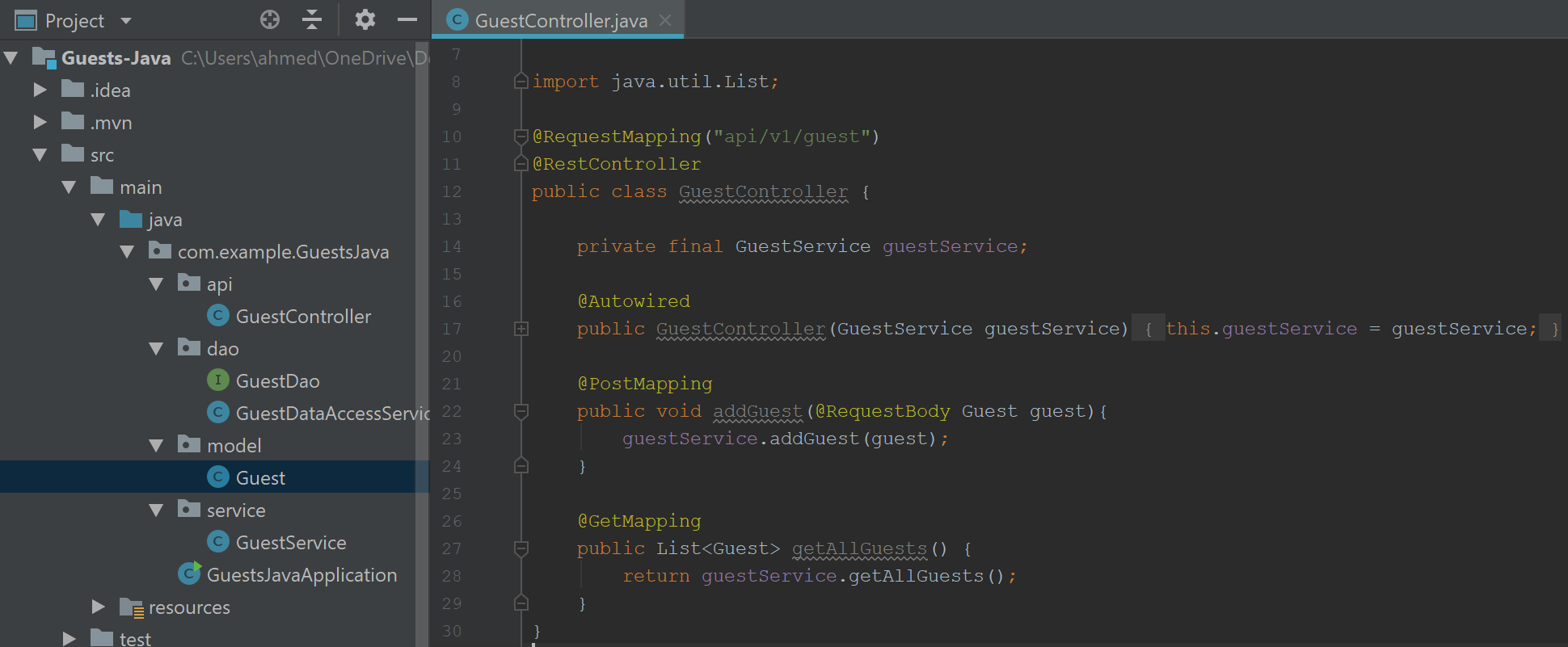




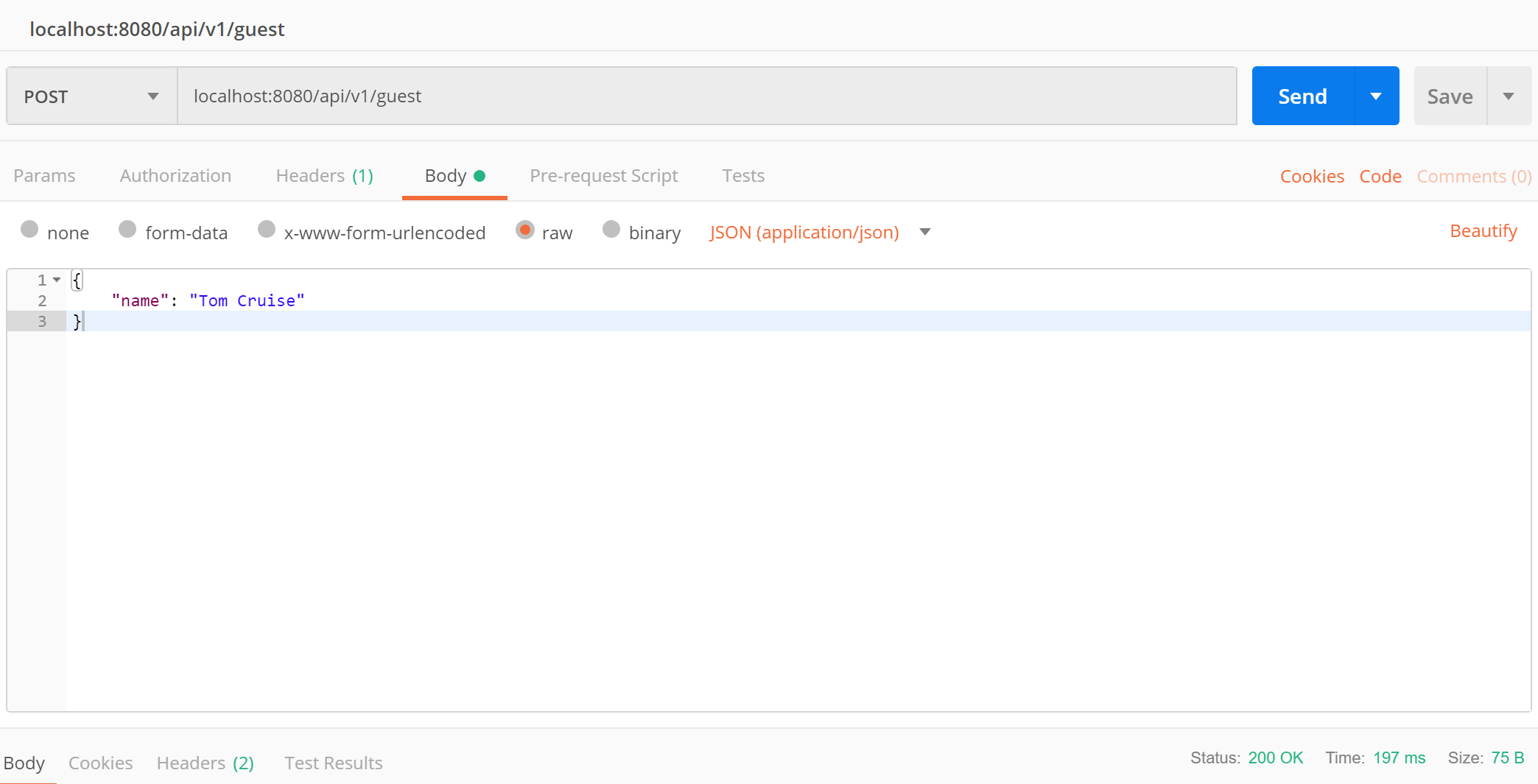
**Step 44:** Return the **DB** instead of **null**;



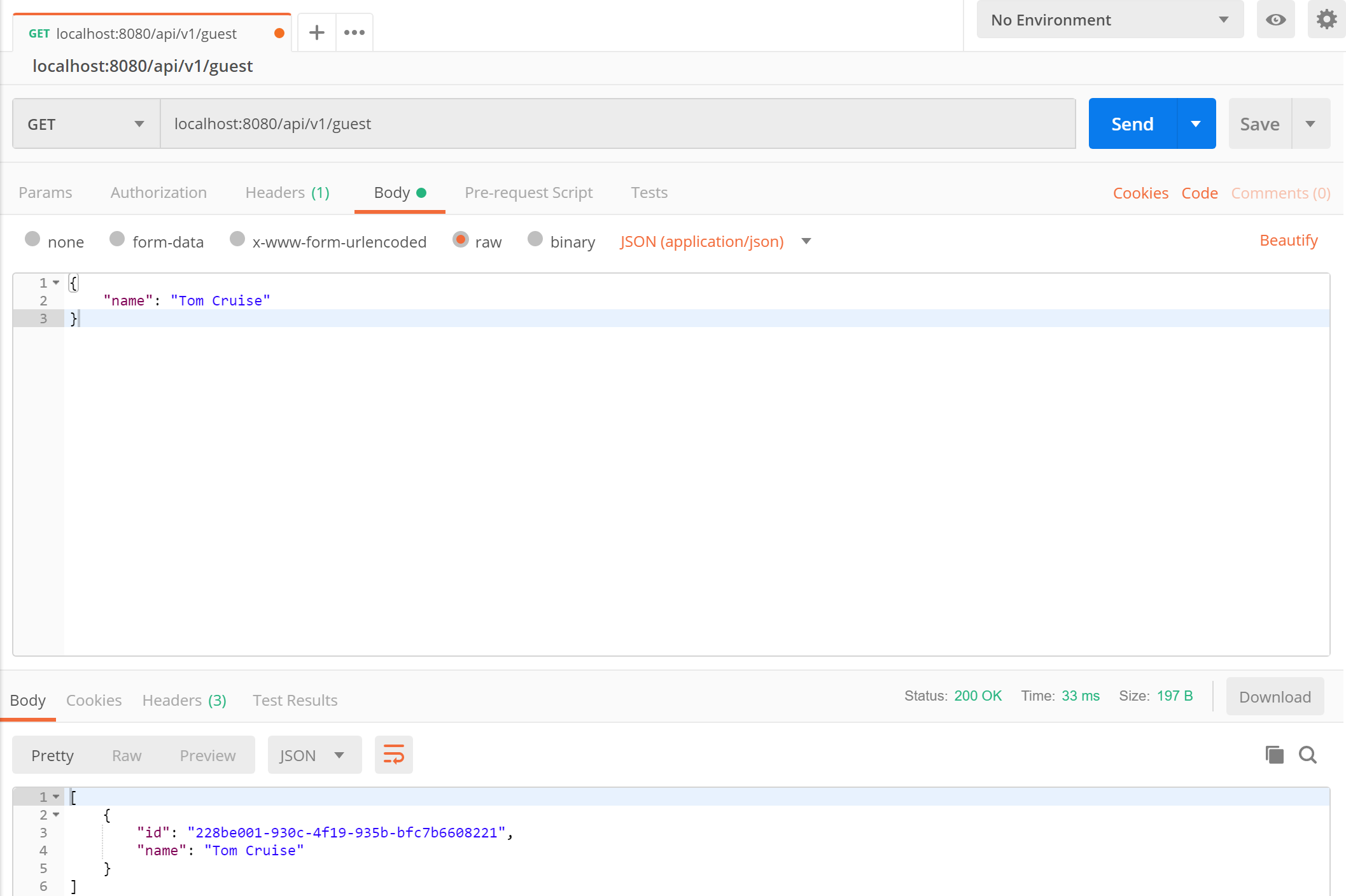
**Step 45:** Create a Method to **Get** all Guests. (Use **GetMapping** Annotation to mark as a Get request.)



**Step 46:** Post Using **Postman Software.**



**Step 47:** Get Using **Postman Software.**



**Note:** Post and Get Requests Working in Java Spring Application.

**UUID Generated.**