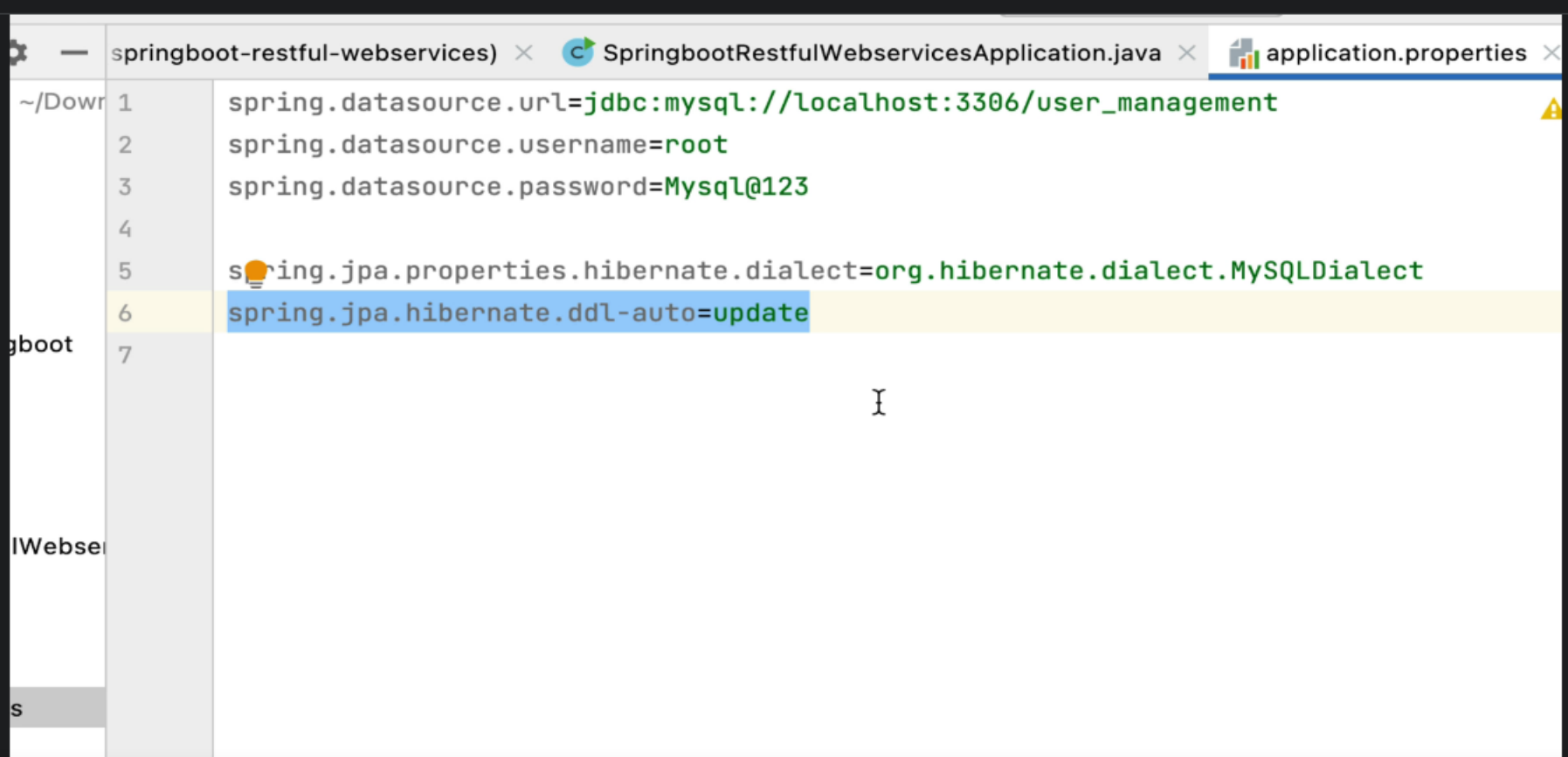
When the `@SpringBootApplication` class runs, it initializes the Spring container and scans for components, configurations, and beans defined in your application. Here's how it works step by step:

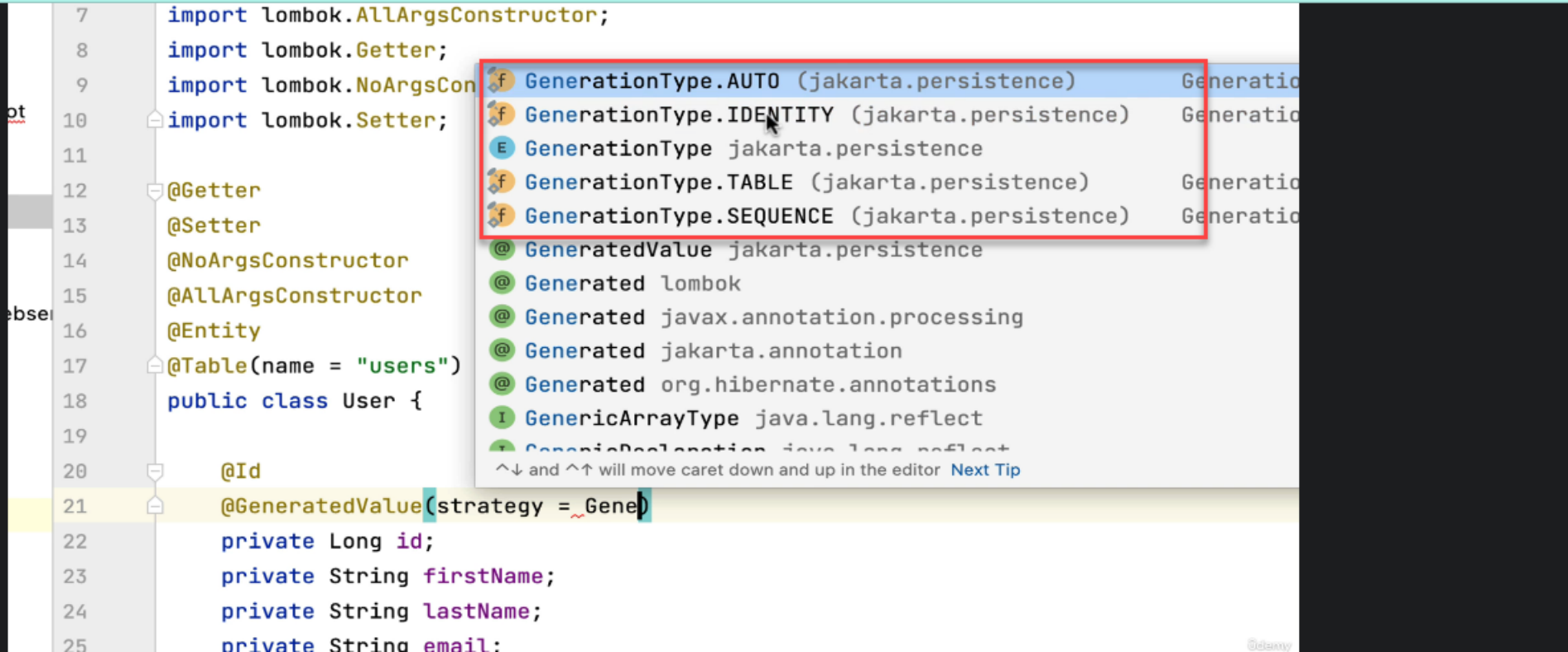
1. **Component Scanning**:
   The `@SpringBootApplication` annotation enables component scanning by default. Spring scans the package containing the main application class and its sub-packages for classes annotated with `@Component`, `@Service`, `@Repository`, `@Controller`, or `@RestController`.
2. **Bean Discovery**:
   During the scan, any method or class annotated with `@Bean` or a stereotype annotation like `@Component` is registered as a bean in the Spring context.
3. **Dependency Injection**:
   Spring resolves dependencies by injecting the appropriate beans wherever required. For example, beans are injected into fields, constructor parameters, or methods annotated with `@Autowired` or via constructor injection.
4. **Runtime Execution**:
   When your `SpringBootApplication` runs, it initializes the Spring Application Context and instantiates the discovered beans. The context manages their lifecycle, and they are ready for use.

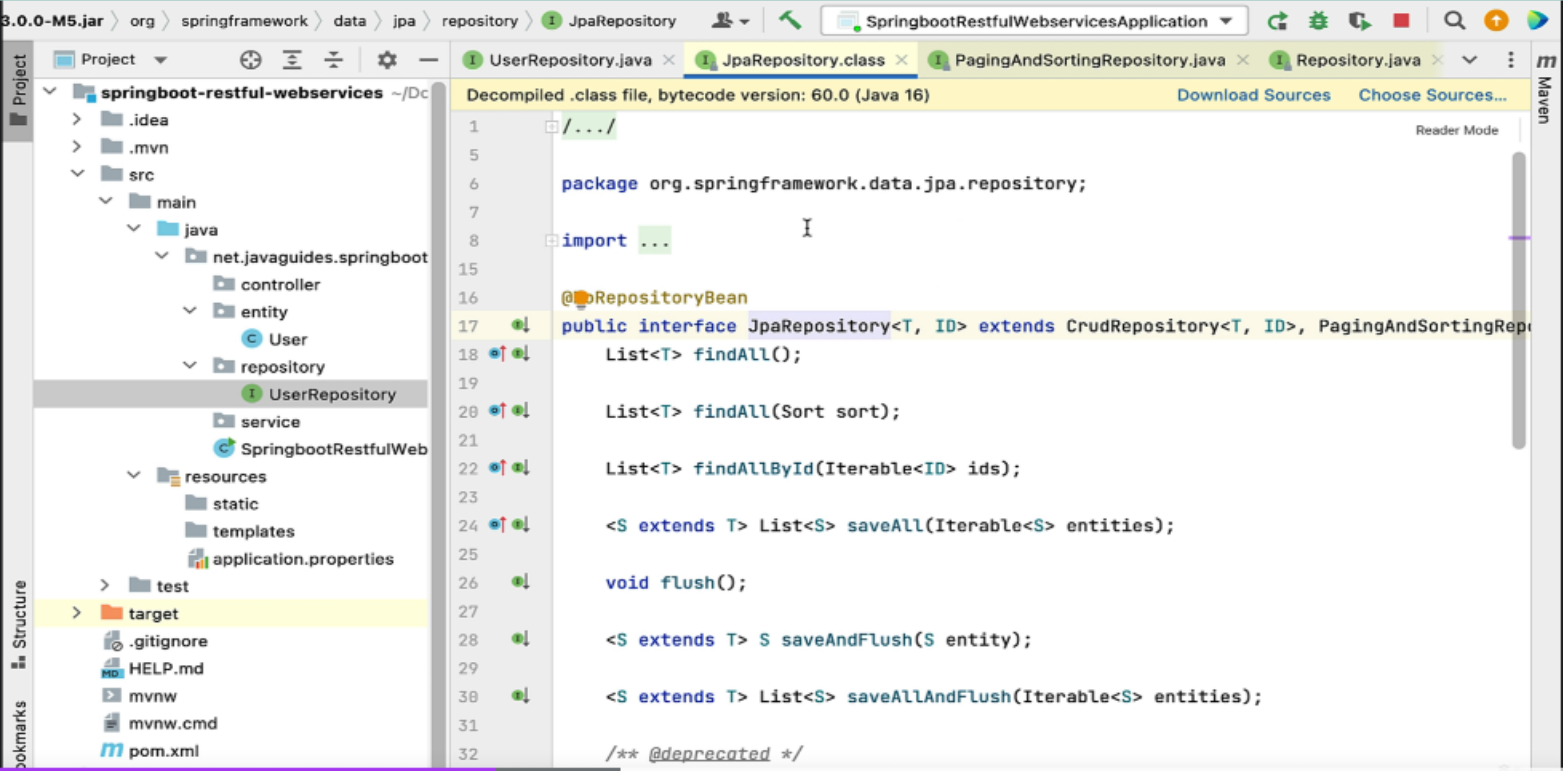
If you had a simple `@Bean` that returned, say, a `String`, and a `@Component` or `@Service` that used it, Spring would wire them together and print the output when invoked.

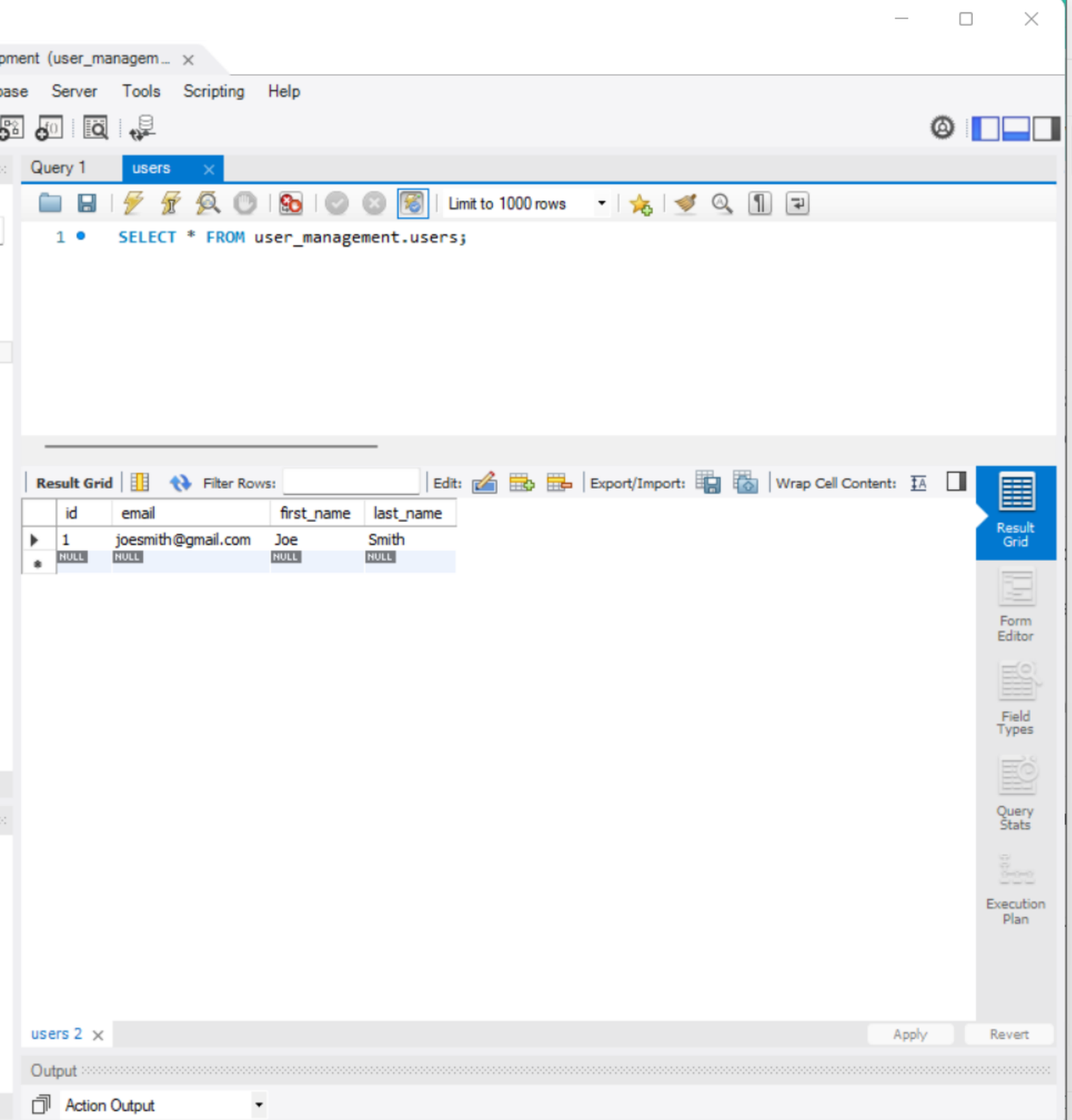
```java
package net.javaguides.springboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;

@Configuration  no usages  kashanuk
public class DatabaseConnectionTest {

    @Autowired  1 usage
    private DataSource dataSource;

    @Bean  no usages  kashanuk
    public CommandLineRunner testConnection() {
        return String[] args -> {
            try (Connection connection = dataSource.getConnection()) {
                System.out.println("Connection Established: " + connection.getCatalog());
            } catch (SQLException e) {
                System.err.println("Failed to establish connection: " + e.getMessage());
            }
        };
    }
}
```

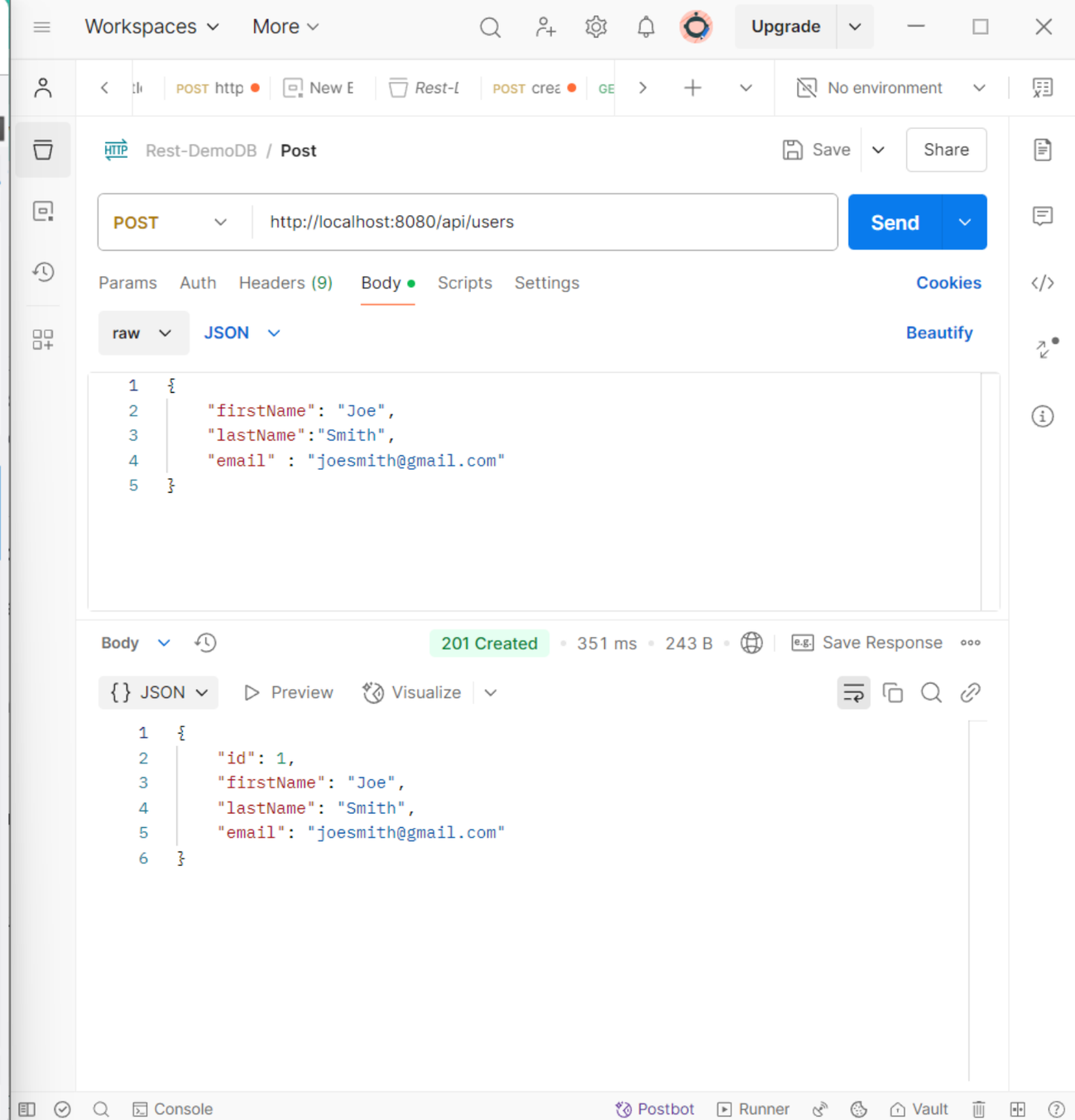

springboot-restful-webservices)
SpringbootRestfulWebservicesApplication.java
application.properties
spring.datasource.url=jdbc:mysql://localhost:3306/user_management
spring.datasource.username=root
spring.datasource.password=Mysql@123
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update

```java
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "first_name", nullable = false)
    private String firstName;
    private String lastName;
    private String email;
}
```


import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsCon
import lombok.Setter;
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "users")
public class User {
@Id
@GeneratedValue(strategy = Gene)
private Long id;
private String firstName;
private String lastName;
private String email;
GenerationType.AUTO (jakarta.persistence)
GenerationType.IDENTITY (jakarta.persistence)
GenerationType jakarta.persistence
GenerationType.TABLE (jakarta.persistence)
GenerationType.SEQUENCE (jakarta.persistence)
GeneratedValue jakarta.persistence
Generated lombok
Generated javax.annotation.processing
Generated jakarta.annotation
Generated org.hibernate.annotations
GenericArrayType java.lang.reflect
^↓ and ^↑ will move caret down and up in the editor Next Tip

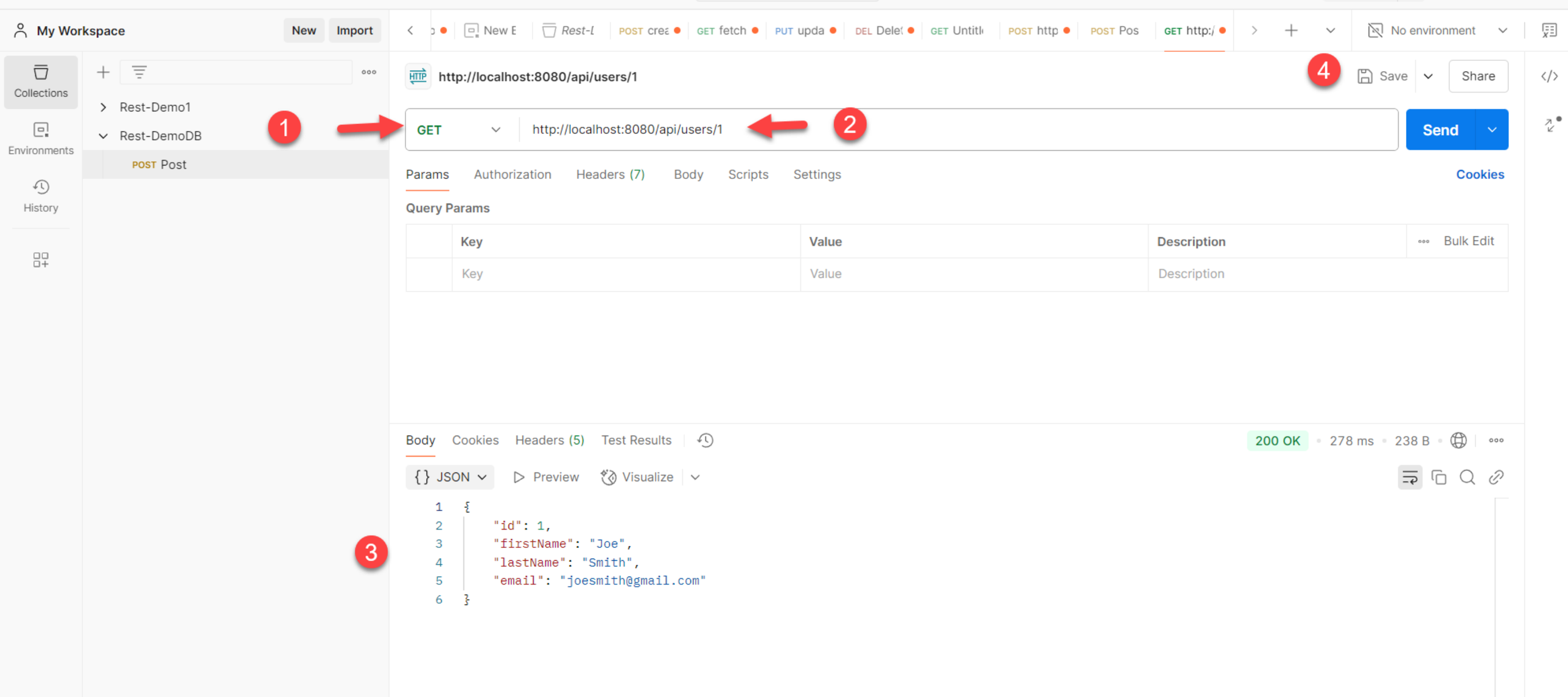

3.0.0-M5.jar 〉 org 〉 springframework 〉 data 〉 jpa 〉 repository 〉 JpaRepository
SpringbootRestfulWebservicesApplication
Project
springboot-restful-webservices ~/Dc
.idea
.mvn
src
main
java
net.javaguides.springboot
controller
entity
User
repository
UserRepository
service
SpringbootRestfulWeb
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
UserRepository.java
JpaRepository.class
PagingAndSortingRepository.java
Repository.java
Decompiled .class file, bytecode version: 60.0 (Java 16)
Download Sources
Choose Sources...
Reader Mode
/.../
package org.springframework.data.jpa.repository;
import ...
@RepositoryBean
public interface JpaRepository<T, ID> extends CrudRepository<T, ID>, PagingAndSortingRep
List<T> findAll();
List<T> findAll(Sort sort);
List<T> findAllById(Iterable<ID> ids);
<S extends T> List<S> saveAll(Iterable<S> entities);
void flush();
<S extends T> S saveAndFlush(S entity);
<S extends T> List<S> saveAllAndFlush(Iterable<S> entities);
/** @deprecated */
Maven
Structure
Bookmarks

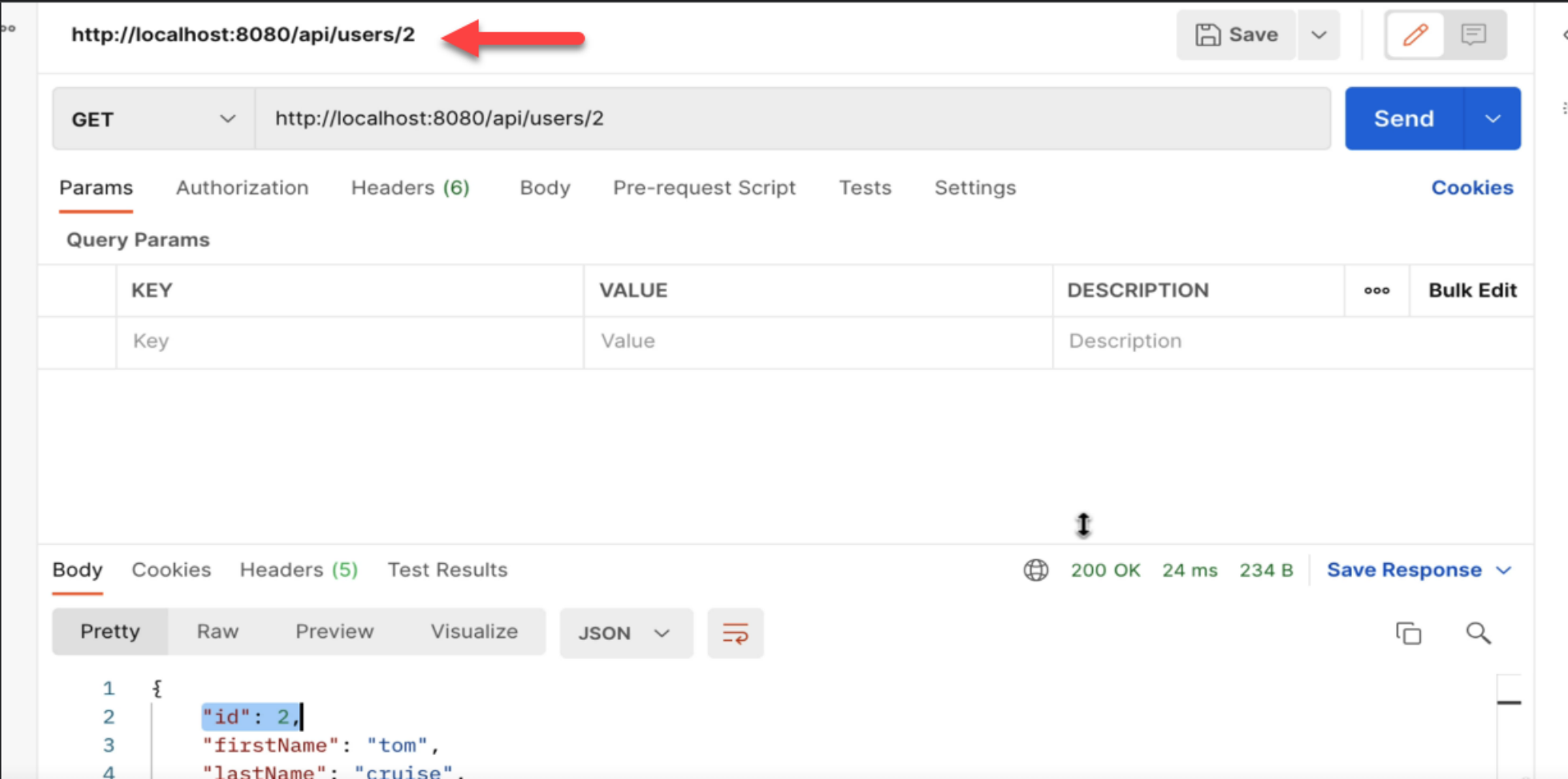

pment (user_managem...
base Server Tools Scripting Help
Query 1
users
Limit to 1000 rows
1 • SELECT * FROM user_management.users;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id email first_name last_name
1 joesmith@gmail.com Joe Smith
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
users 2
Apply
Revert
Output
Action Output

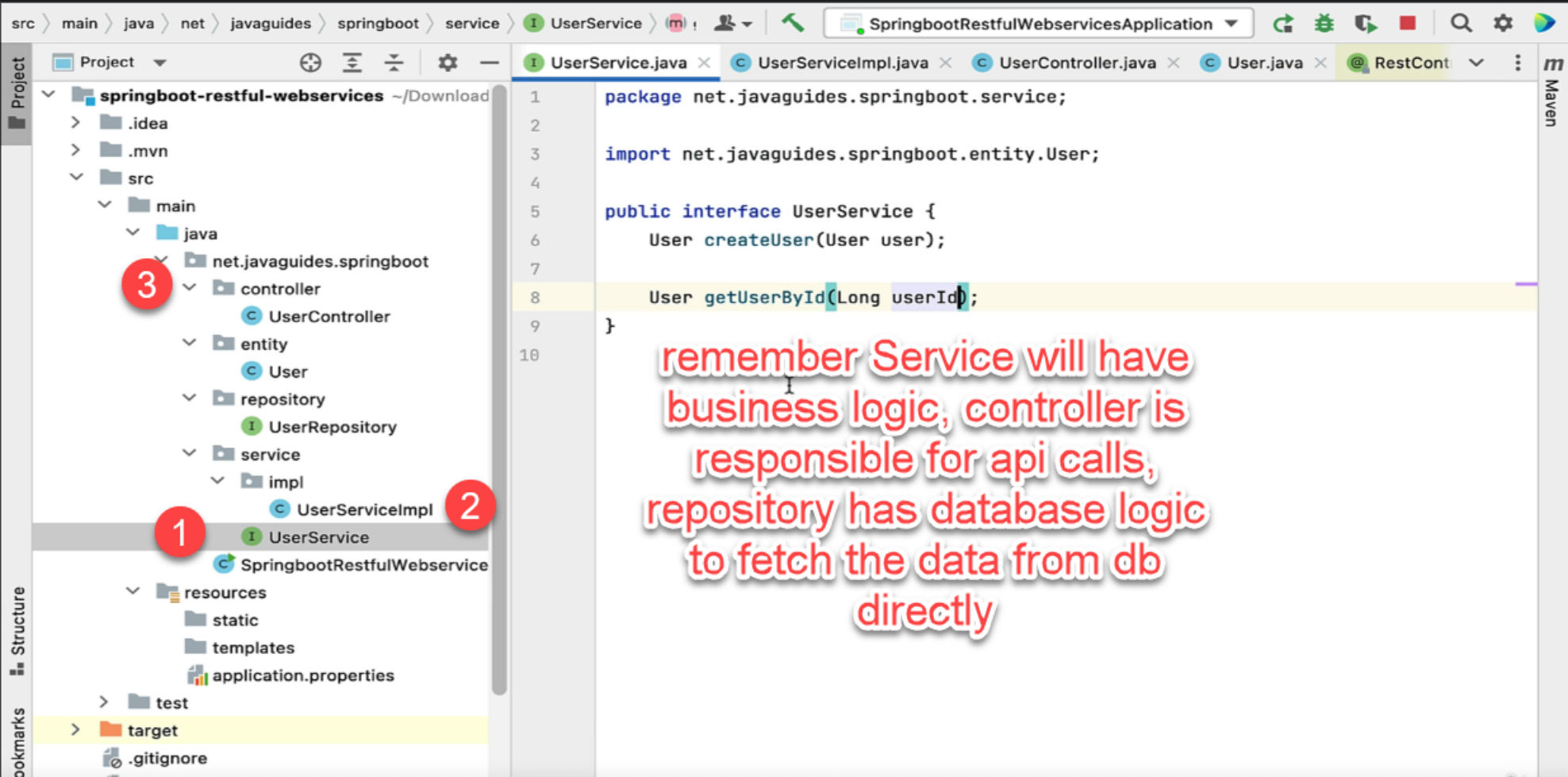

Workspaces
More
Upgrade
No environment
Rest-DemoDB / Post
Save
Share
POST
http://localhost:8080/api/users
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw
JSON
Beautify
{
"firstName": "Joe",
"lastName":"Smith",
"email" : "joesmith@gmail.com"
}
Body
201 Created
351 ms
243 B
Save Response
JSON
Preview
Visualize
{
"id": 1,
"firstName": "Joe",
"lastName": "Smith",
"email": "joesmith@gmail.com"
}
Console
Postbot
Runner
Vault

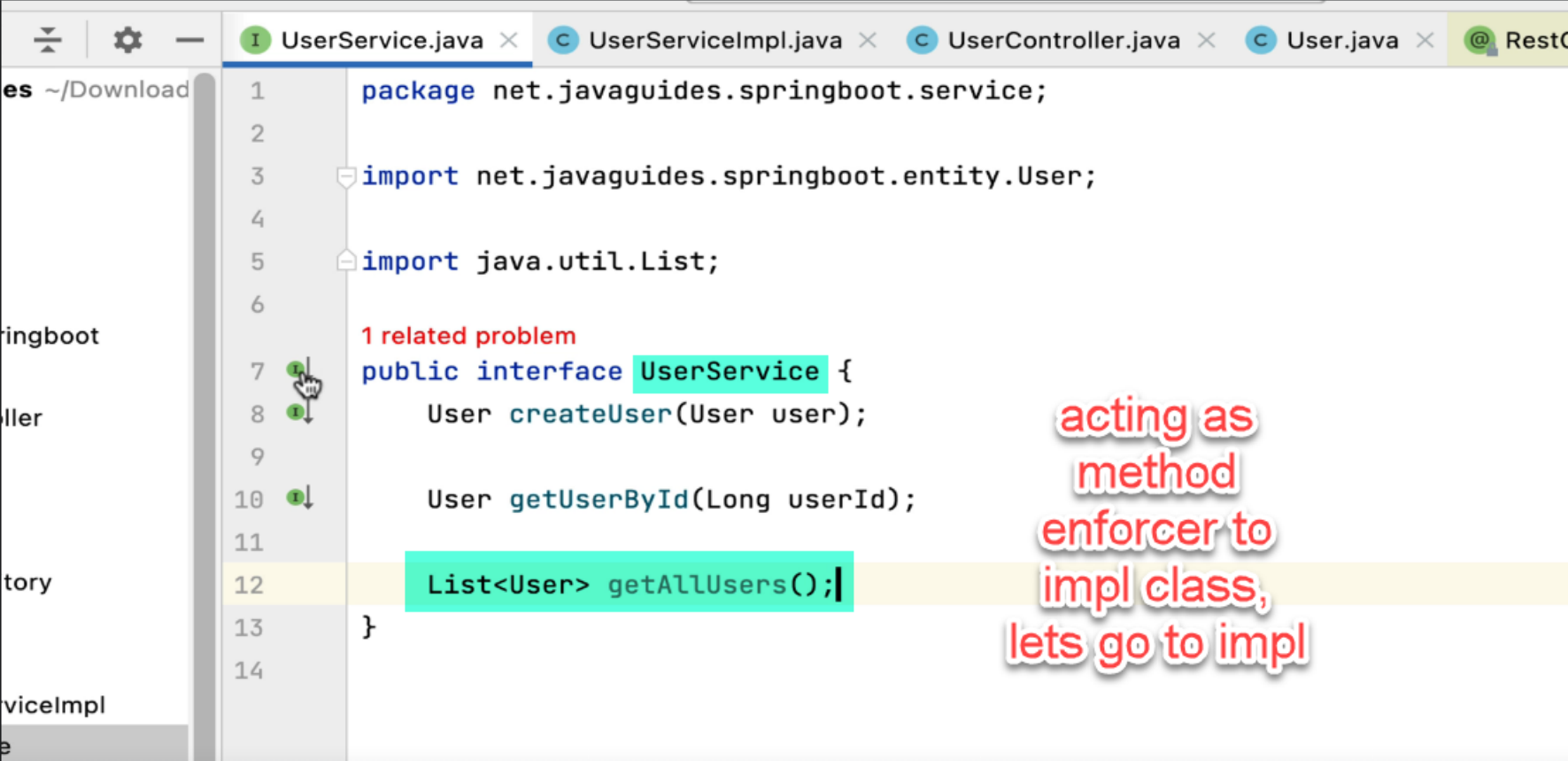

UserService.java
UserServiceImpl.java
UserController.java
User.java
RestController.class
ngboot-restful-webservices ~/
idea
mvn
rc
main
java
net.javaguides.springbo
controller
UserController
entity
User
repository
UserRepository
service
impl
UserServiceIn
UserService
SpringbootRestfulW
resources
static
templates
application.properties
test
arget
3
1
2
7 import org.springframework.stereotype.Service;
8
9 import java.util.Optional;
10
11 @Service
12 @AllArgsConstructor
13 public class UserServiceImpl implements UserService {
14
15 private UserRepository userRepository;
16
17 @Override
18 public User createUser(User user) {
19 return userRepository.save(user);
20 }
21
22 @Override
23 public User getUserById(Long userId) {
24 Optional<User> optionalUser = userRepository.findById(userId);
25 return optionalUser.get();
26 }
27 }
28
Maven


main
java
net.javaguides.springb
controller
UserController
entity
User
repository
UserRepository
service
impl
UserService
UserService
SpringbootRestfulV
resources
static
templates
application.properties
test
target
Structure
kmarks
// build create User REST API
@PostMapping
public ResponseEntity<User> createUser(@RequestBody User user){
User savedUser = userService.createUser(user);
return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
}
// build get user by id REST API
// http://localhost:8080/api/users/1
@GetMapping("{id}")
public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
User user = userService.getUserById(userId);
return new ResponseEntity<>(user, HttpStatus.OK);
}
}
URI TEMPLATE
VARIABLE


My Workspace
New
Import
Collections
Environments
History
Rest-Demo1
Rest-DemoDB
POST Post
1
GET
http://localhost:8080/api/users/1
2
Send
http://localhost:8080/api/users/1
4
Save
Share
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
278 ms
238 B
JSON
Preview
Visualize
3
{
"id": 1,
"firstName": "Joe",
"lastName": "Smith",
"email": "joesmith@gmail.com"
}

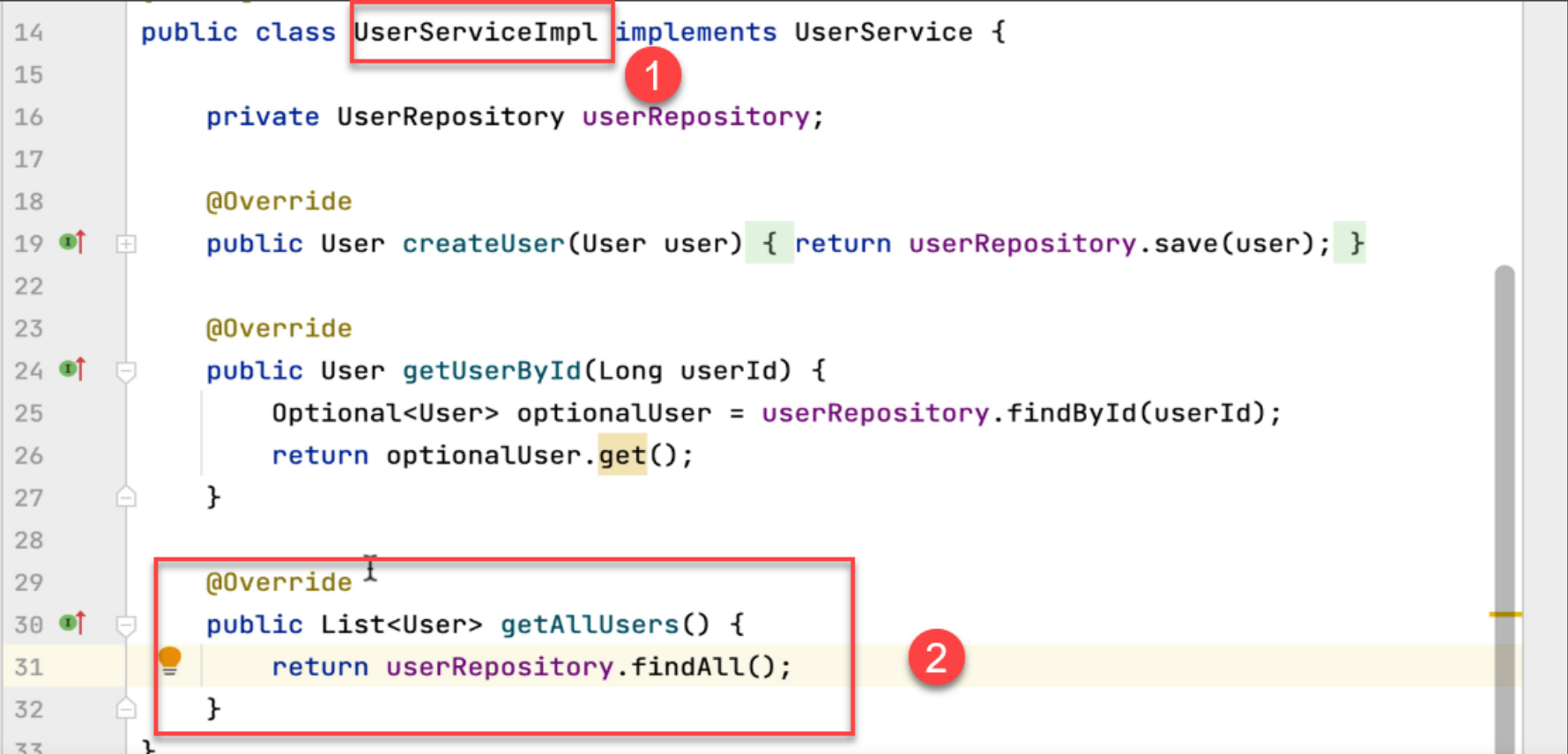

http://localhost:8080/api/users/2
Save
GET
http://localhost:8080/api/users/2
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
200 OK
24 ms
234 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 2,
"firstName": "tom",
"lastName": "cruise",

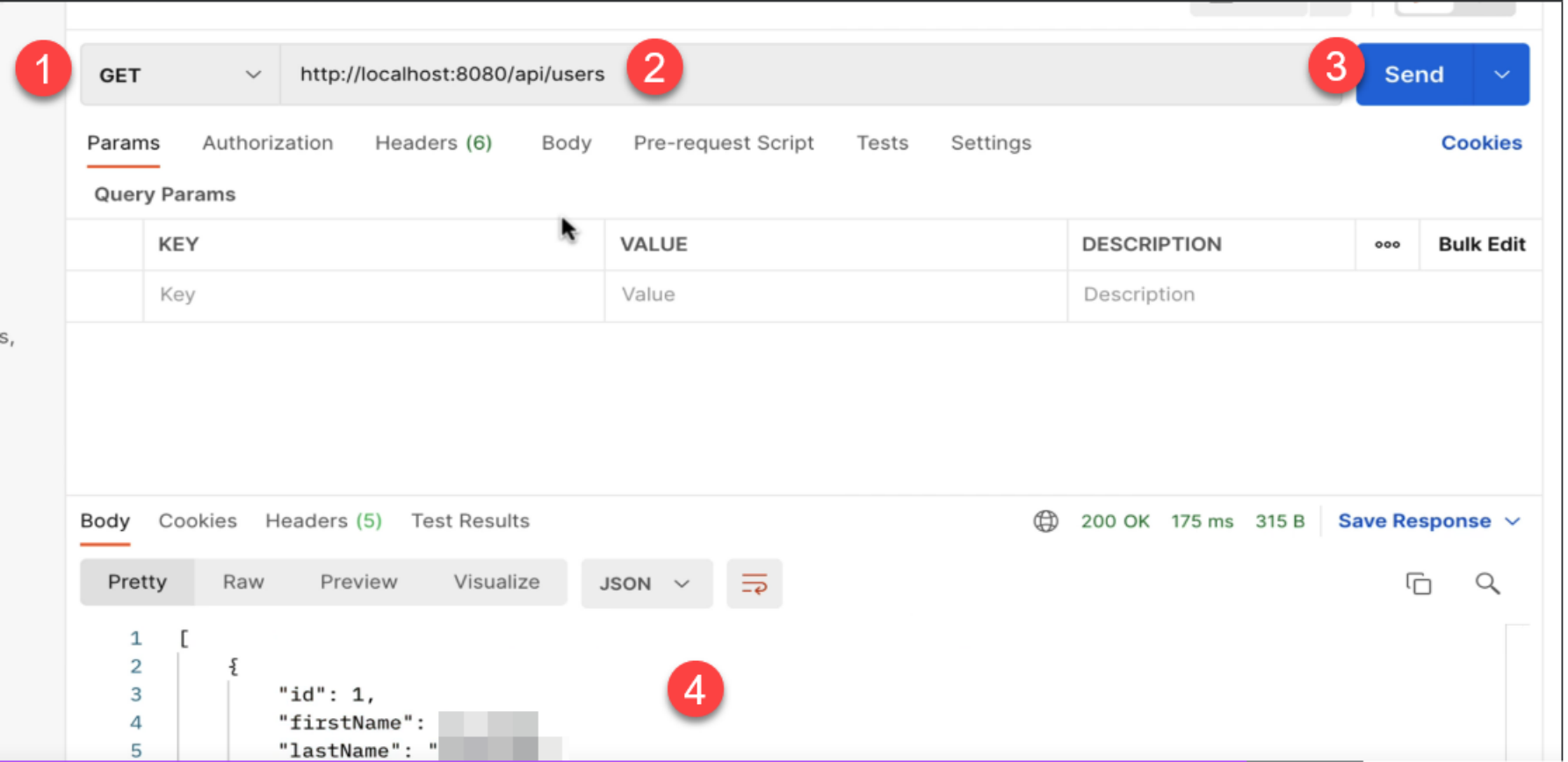

src main java net javaguides springboot service UserService
SpringbootRestfulWebservicesApplication
Project
springboot-restful-webservices ~/Download
.idea
.mvn
src
main
java
net.javaguides.springboot
3
controller
UserController
entity
User
repository
UserRepository
service
impl
UserServiceImpl
2
1
UserService
SpringbootRestfulWebservice
resources
static
templates
application.properties
test
target
.gitignore
UserService.java
UserServiceImpl.java
UserController.java
User.java
RestCont
package net.javaguides.springboot.service;
import net.javaguides.springboot.entity.User;
public interface UserService {
User createUser(User user);
User getUserById(Long userId);
}
remember Service will have business logic, controller is responsible for api calls, repository has database logic to fetch the data from db directly
Project
Structure
Bookmarks
Maven


UserService.java
UserServiceImpl.java
UserController.java
User.java
1 package net.javaguides.springboot.service;
2
3 import net.javaguides.springboot.entity.User;
4
5 import java.util.List;
6
1 related problem
7 public interface UserService {
8 User createUser(User user);
9
10 User getUserById(Long userId);
11
12 List<User> getAllUsers();
13 }
14
acting as method enforcer to impl class, lets go to impl

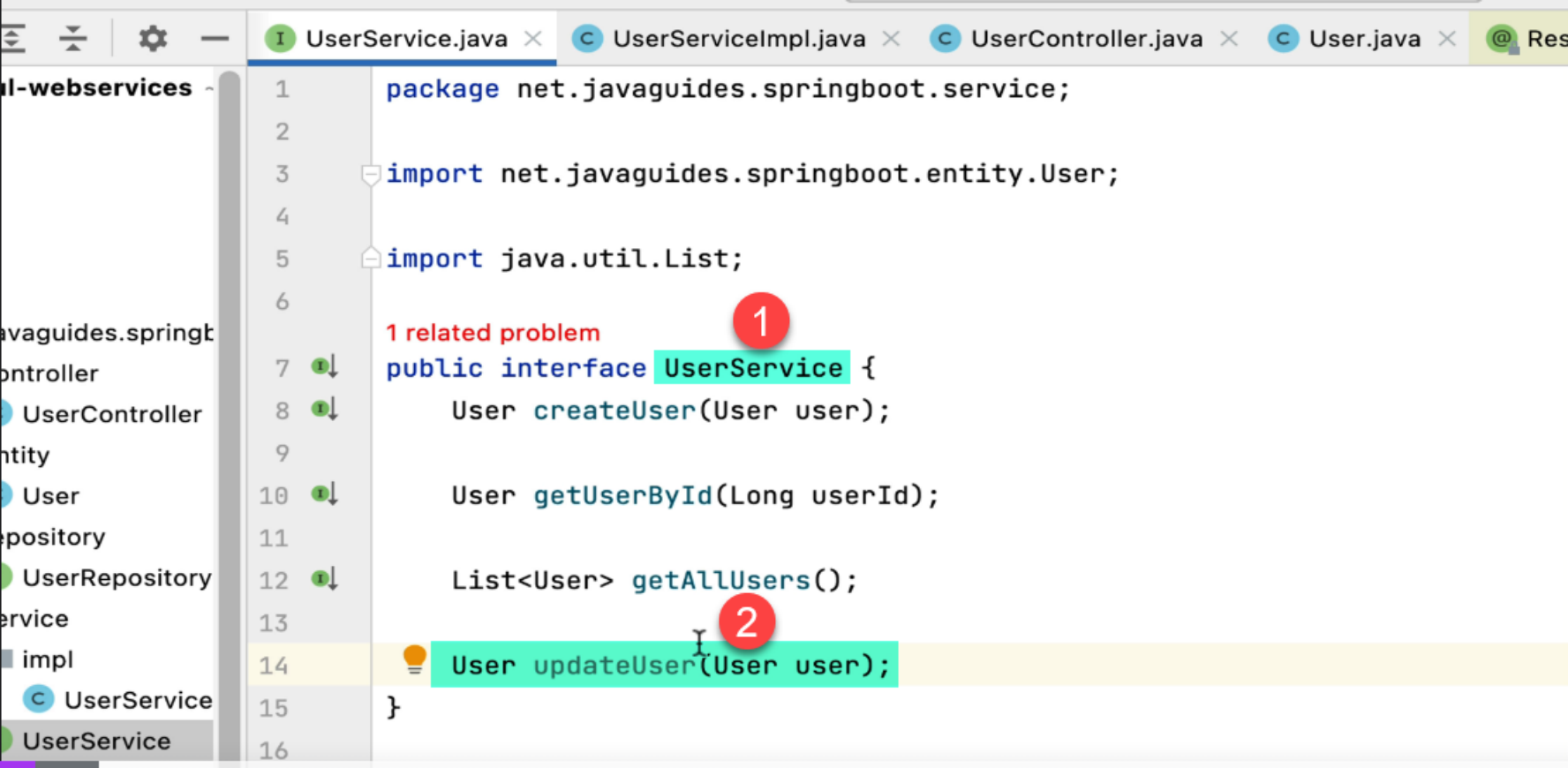
```
22          User savedUser = userService.createUser(user);
23          return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
24      }
25
26      // build get user by id REST API
27      // http://localhost:8080/api/users/1
28      @GetMapping("{id}")
29      public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
30          User user = userService.getUserById(userId);
31          return new ResponseEntity<>(user, HttpStatus.OK);
32      }
33
34      // Build Get All Users REST API
35      // http://localhost:8080/api/users
36      @GetMapping
37      public ResponseEntity<List<User>> getAllUsers(){
38          List<User> users = userService.getAllUsers();
39          return new ResponseEntity<>(users, HttpStatus.OK);
40      }
```

in the controller


public class UserServiceImpl implements UserService {
1
private UserRepository userRepository;
@Override
public User createUser(User user) { return userRepository.save(user); }
@Override
public User getUserById(Long userId) {
Optional<User> optionalUser = userRepository.findById(userId);
return optionalUser.get();
}
@Override
public List<User> getAllUsers() {
return userRepository.findAll();
2
}



you can post 1 more user and then call that user using get all or specific id


UserService.java
UserServiceImpl.java
UserController.java
User.java
l-webservices
avaguides.springb
ontroller
UserController
ntity
User
epository
UserRepository
ervice
impl
UserService
UserService
1
package net.javaguides.springboot.service;
import net.javaguides.springboot.entity.User;
import java.util.List;
1 related problem
public interface UserService {
User createUser(User user);
User getUserById(Long userId);
List<User> getAllUsers();
2
User updateUser(User user);
}

```
24      public User getUserById(Long userId) {
25          Optional<User> optionalUser = userRepository.findById(userId);
26          return optionalUser.get();
27      }
28
29      @Override
30      public List<User> getAllUsers() {
31          return userRepository.findAll();
32      }
33
34      @Override
35      public User updateUser(User user) {
36          
37          return null;
38      }
39  }
40
```

guess which class it is UserServiceImpl

first we will get the use and then will do the update

```
28          Optional<User> optionalUser = userRepository.findById(userId);
29          return optionalUser.get();
30      }
31
32      @Override
33      public List<User> getAllUsers() {
34          return userRepository.findAll();
35      }
36
37      @Override
38      public User updateUser(User user) {
39          User existingUser = userRepository.findById(user.getId()).get();
40          existingUser.setFirstName(user.getFirstName());
41          existingUser.setLastName(user.getLastName());
42          existingUser.setEmail(user.getEmail());
43          return null;
44      }
```

updating from the user object to existing user object to bring it available for update

```
28          Optional<User> optionalUser = userRepository.findById(userId);
29          return optionalUser.get();
30      }
31
32      @Override
33      public List<User> getAllUsers() {
34          return userRepository.findAll();
35      }
36
37      @Override
38      public User updateUser(User user) {
39          User existingUser = userRepository.findById(user.getId()).get();
40          existingUser.setFirstName(user.getFirstName());
41          existingUser.setLastName(user.getLastName());
42          existingUser.setEmail(user.getEmail());
43          return null;
44      }
```

next we are going to save this user into existing database and recall which layer is responsible for saving in the database ?

repository

```
31          return new ResponseEntity<>(user, HttpStatus.OK);
32      }
33
34      // Build Get All Users REST API
35      // http://localhost:8080/api/users
36      @GetMapping
37      public ResponseEntity<List<User>> getAllUsers(){
38          List<User> users = userService.getAllUsers();
39          return new ResponseEntity<>(users, HttpStatus.OK);
40      }
41
42      // Build Update User REST API
43      public ResponseEntity<User> updateUser(User user){
44          User updatedUser = userService.updateUser(user);
45          return new ResponseEntity<>(updatedUser, HttpStatus.OK);
46      }
47  }
```

2

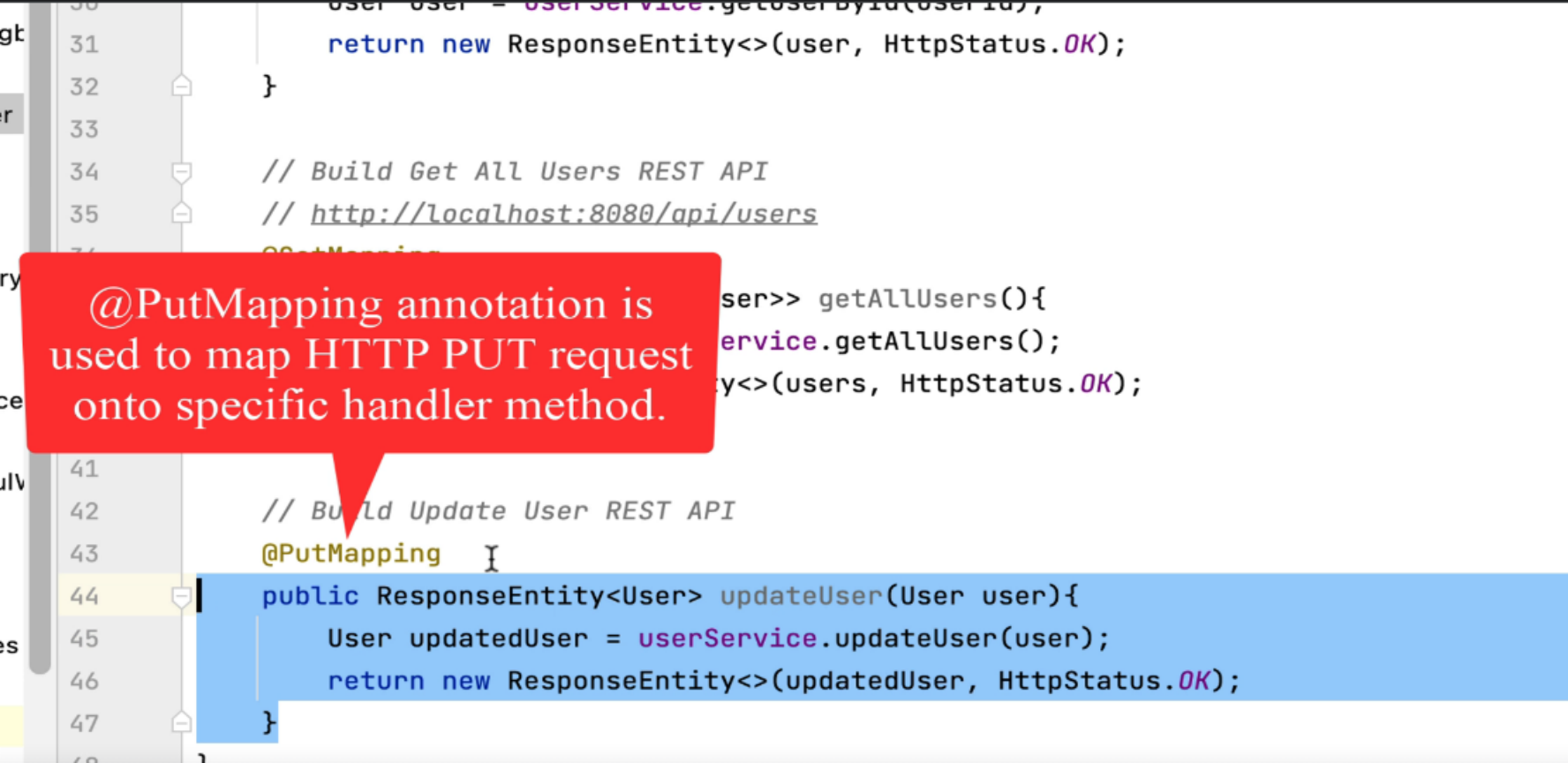
can you guess what we are missing here ?

1

Now in controller we are going to call the method to update user and then controller will let the user know using http reponse that is updated

```
28          Optional<User> optionalUser = userRepository.findById(userId);
29          return optionalUser.get();
30      }
31
32      @Override
33      public List<User> getAllUsers() {
34          return userRepository.findAll();
35      }
36
37      @Override
38      public User updateUser(User user) {
39          User existingUser = userRepository.findById(user.getId()).get();
40          existingUser.setFirstName(user.getFirstName());
41          existingUser.setLastName(user.getLastName());
42          existingUser.setEmail(user.getEmail());
43          User updatedUser = userRepository.save(existingUser);
44          return updatedUser;
```

now we will save it into the database and return the updated user as this method returns the User Object

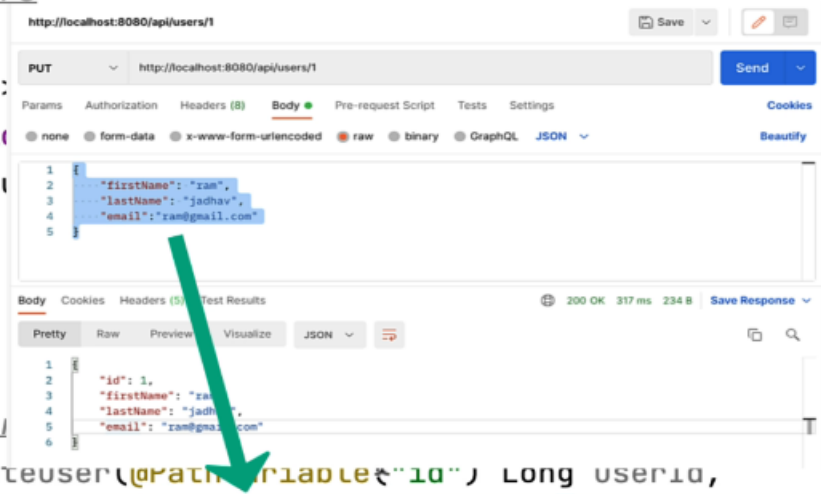



```
        return new ResponseEntity<>(user, HttpStatus.OK);
    }

    // Build Get All Users REST API
    // http://localhost:8080/api/users
    @GetMapping
    public ResponseEntity<List<User>> getAllUsers(){
        List<User> users = userService.getAllUsers();
        return new ResponseEntity<>(users, HttpStatus.OK);
    }

    // Build Update User REST API
    @PutMapping
    public ResponseEntity<User> updateUser(User user){
        User updatedUser = userService.updateUser(user);
        return new ResponseEntity<>(updatedUser, HttpStatus.OK);
    }
}
```

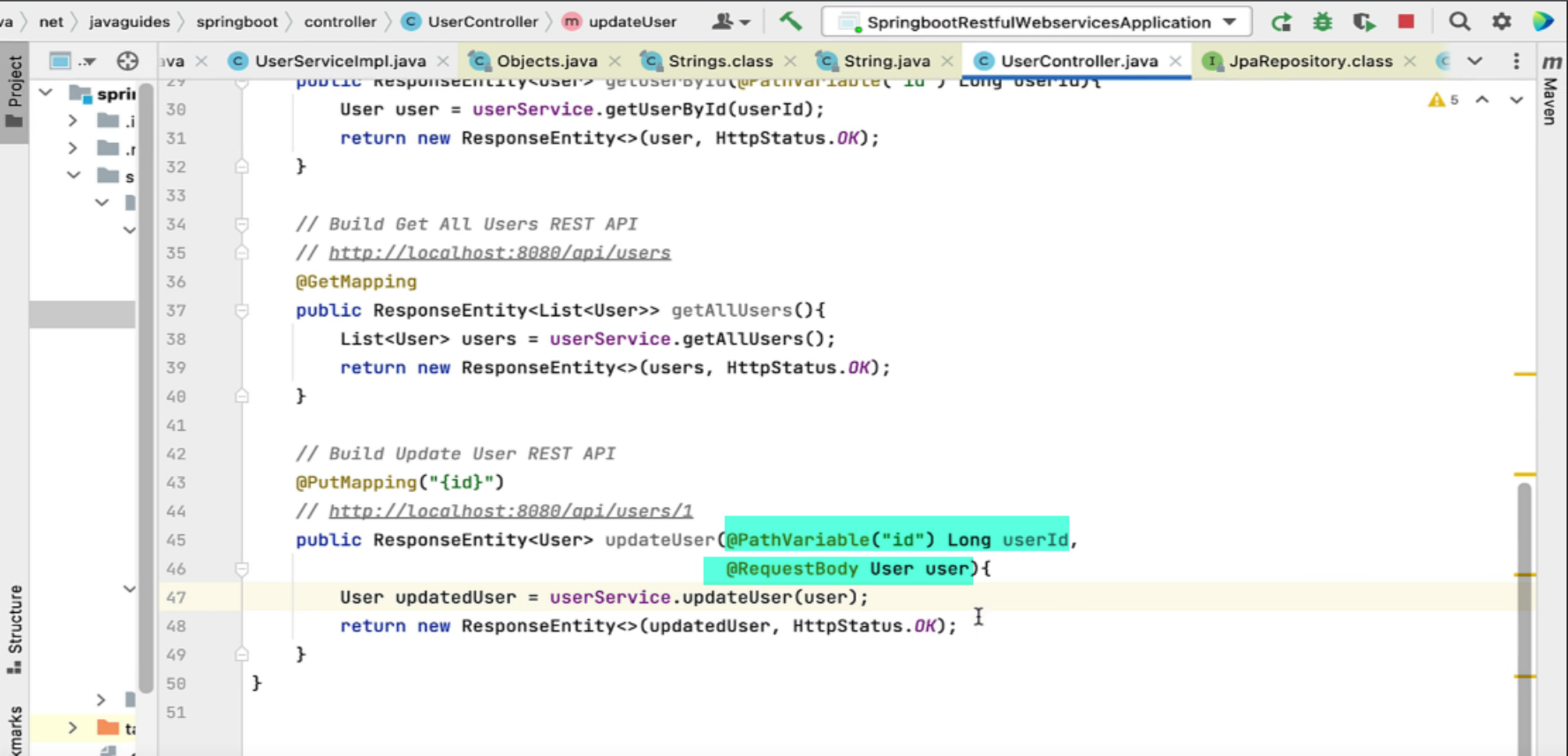
```
35      // http://localhost:8080/api/users
36      @GetMapping
37      public ResponseEntity<List<User>> getAllUsers(){
38          List<User> users = userService.getAllUsers();
39          return new ResponseEntity<>(users, HttpStatus.OK);
40      }
41
42      // Build Update User REST API
43      @PutMapping("{id}")
44      // http://localhost:8080/api/users/1
45      public ResponseEntity<User> updateUser(User user){
46          User updatedUser = userService.updateUser(user);
47          return new ResponseEntity<>(updatedUser, HttpStatus.OK);
48      }
49  }
50
```

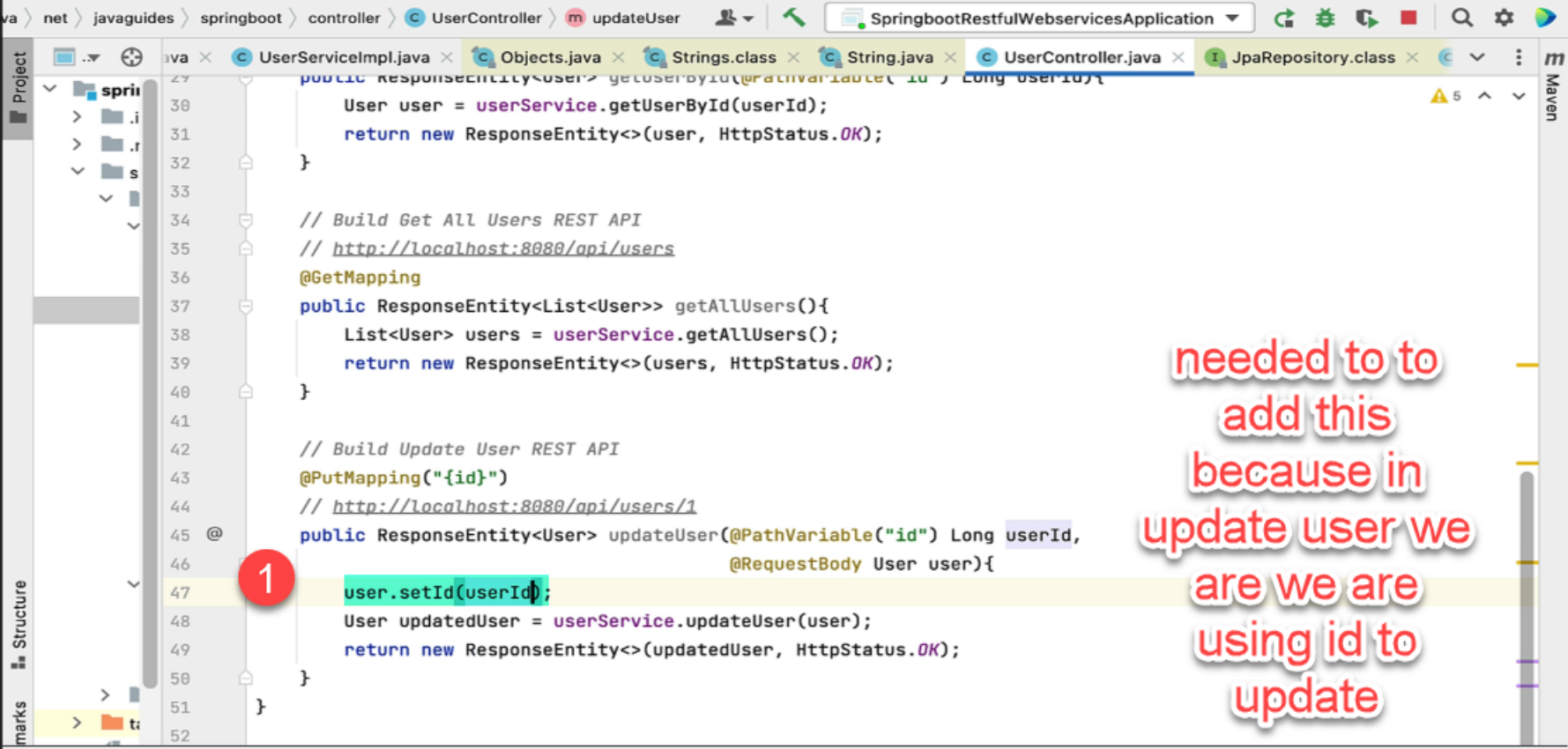
remember we did pathvariable same needs to be done to refresh

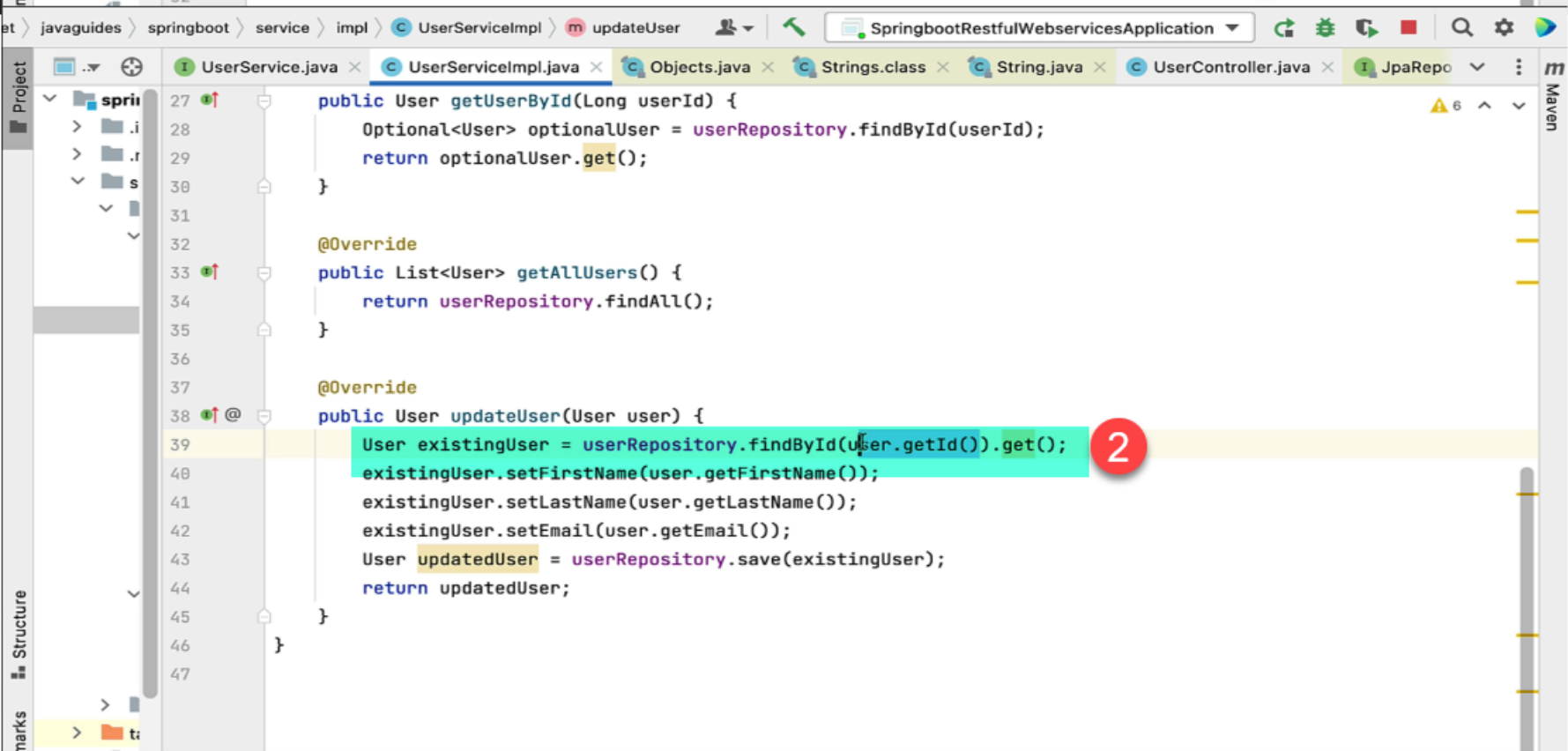
```
://localhost:8080/api/users
oing
ResponseEntity<List<User>>
t<User> users = userServi
urn new ResponseEntity<>(

d Update User REST API
oing("{id}")
://localhost:8080/api/use
ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
                                @RequestBody User user){
r updatedUser = userService.updateUser(user);
urn new ResponseEntity<>(updatedUser, HttpStatus.OK);
```

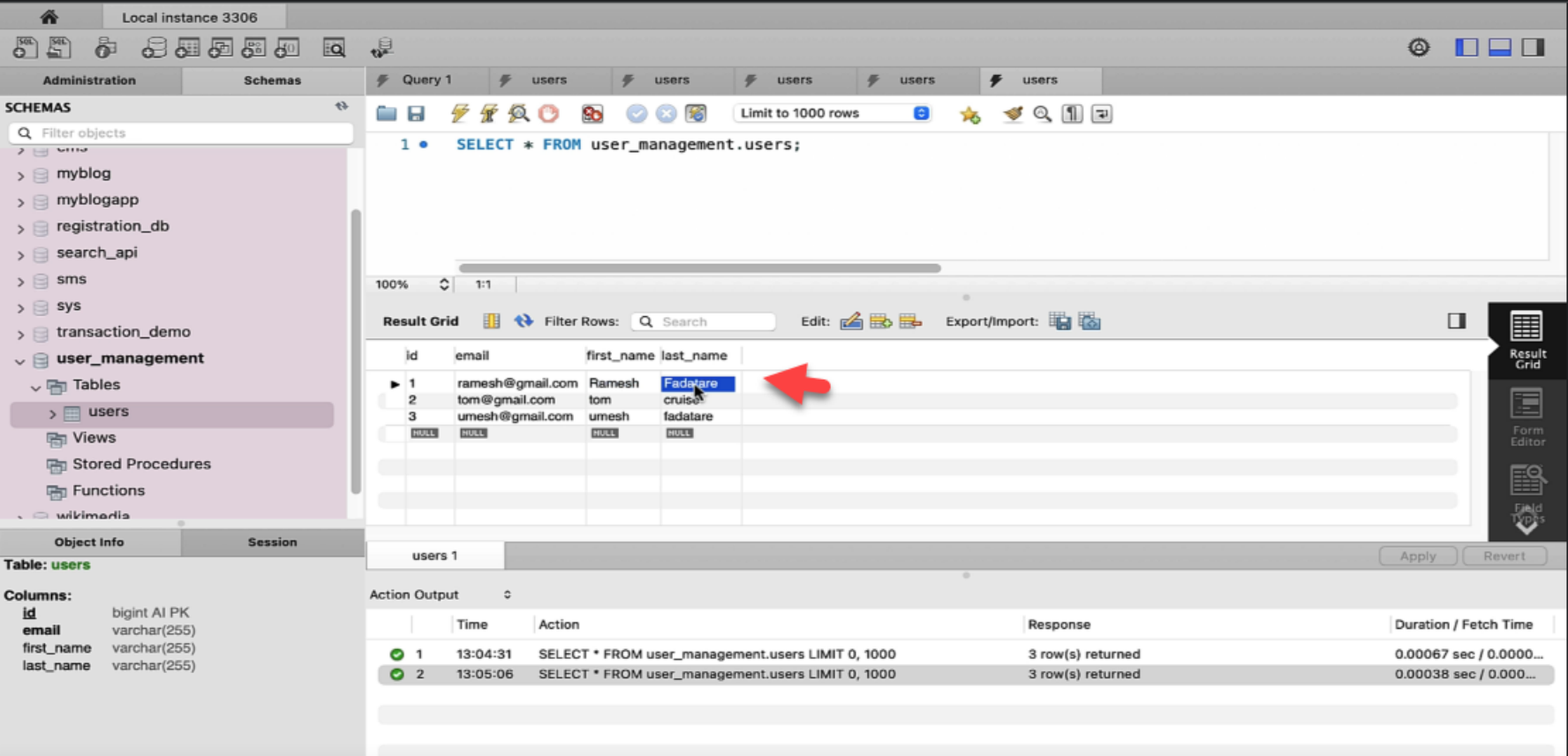


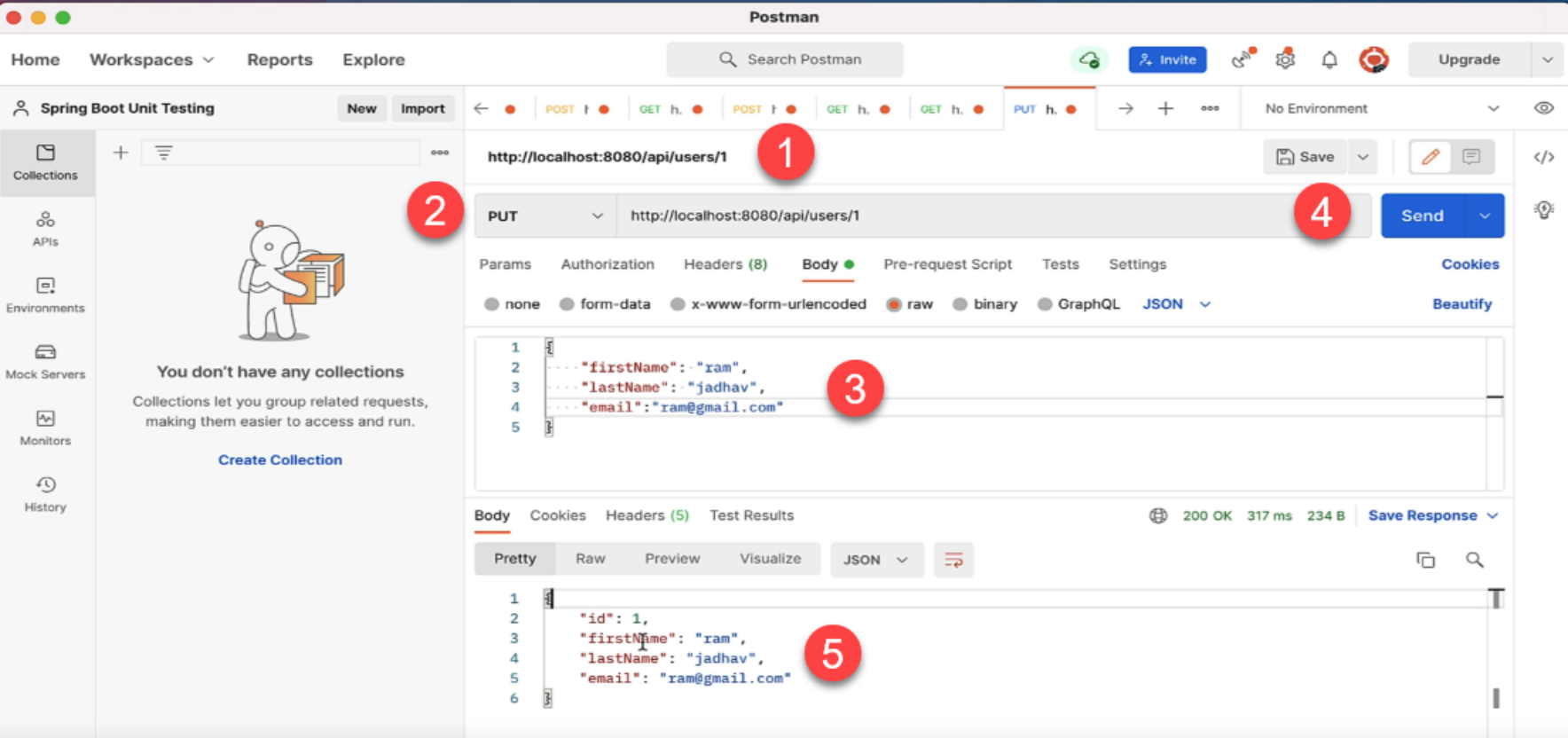
convert json user object to java object

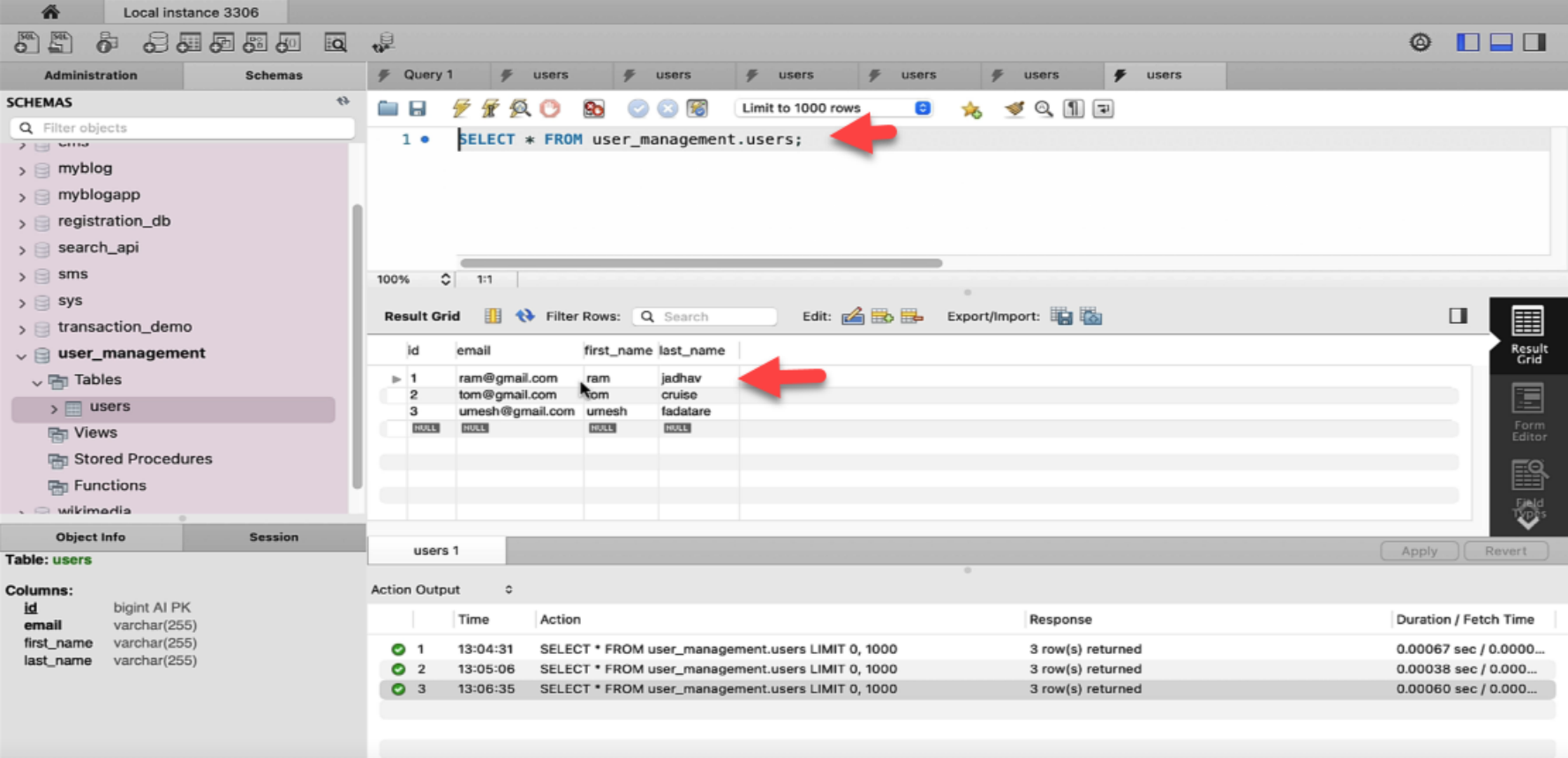

net
javaguides
springboot
controller
UserController
updateUser
SpringbootRestfulWebservicesApplication
UserServiceImpl.java
Objects.java
Strings.class
String.java
UserController.java
JpaRepository.class
Project
Structure
Maven
public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
User user = userService.getUserById(userId);
return new ResponseEntity<>(user, HttpStatus.OK);
}
// Build Get All Users REST API
// http://localhost:8080/api/users
@GetMapping
public ResponseEntity<List<User>> getAllUsers(){
List<User> users = userService.getAllUsers();
return new ResponseEntity<>(users, HttpStatus.OK);
}
// Build Update User REST API
@PutMapping("{id}")
// http://localhost:8080/api/users/1
public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
@RequestBody User user){
User updatedUser = userService.updateUser(user);
return new ResponseEntity<>(updatedUser, HttpStatus.OK);
}
}


net › javaguides › springboot › controller › UserController › updateUser
SpringbootRestfulWebservicesApplication
UserServiceImpl.java
Objects.java
Strings.class
String.java
UserController.java
JpaRepository.class
30 User user = userService.getUserById(userId);
31 return new ResponseEntity<>(user, HttpStatus.OK);
32 }
34 // Build Get All Users REST API
35 // http://localhost:8080/api/users
36 @GetMapping
37 public ResponseEntity<List<User>> getAllUsers(){
38 List<User> users = userService.getAllUsers();
39 return new ResponseEntity<>(users, HttpStatus.OK);
40 }
42 // Build Update User REST API
43 @PutMapping("{id}")
44 // http://localhost:8080/api/users/1
45 public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
46 @RequestBody User user){
47 user.setId(userId);
48 User updatedUser = userService.updateUser(user);
49 return new ResponseEntity<>(updatedUser, HttpStatus.OK);
50 }
51 }
1
needed to to add this because in update user we are we are using id to update

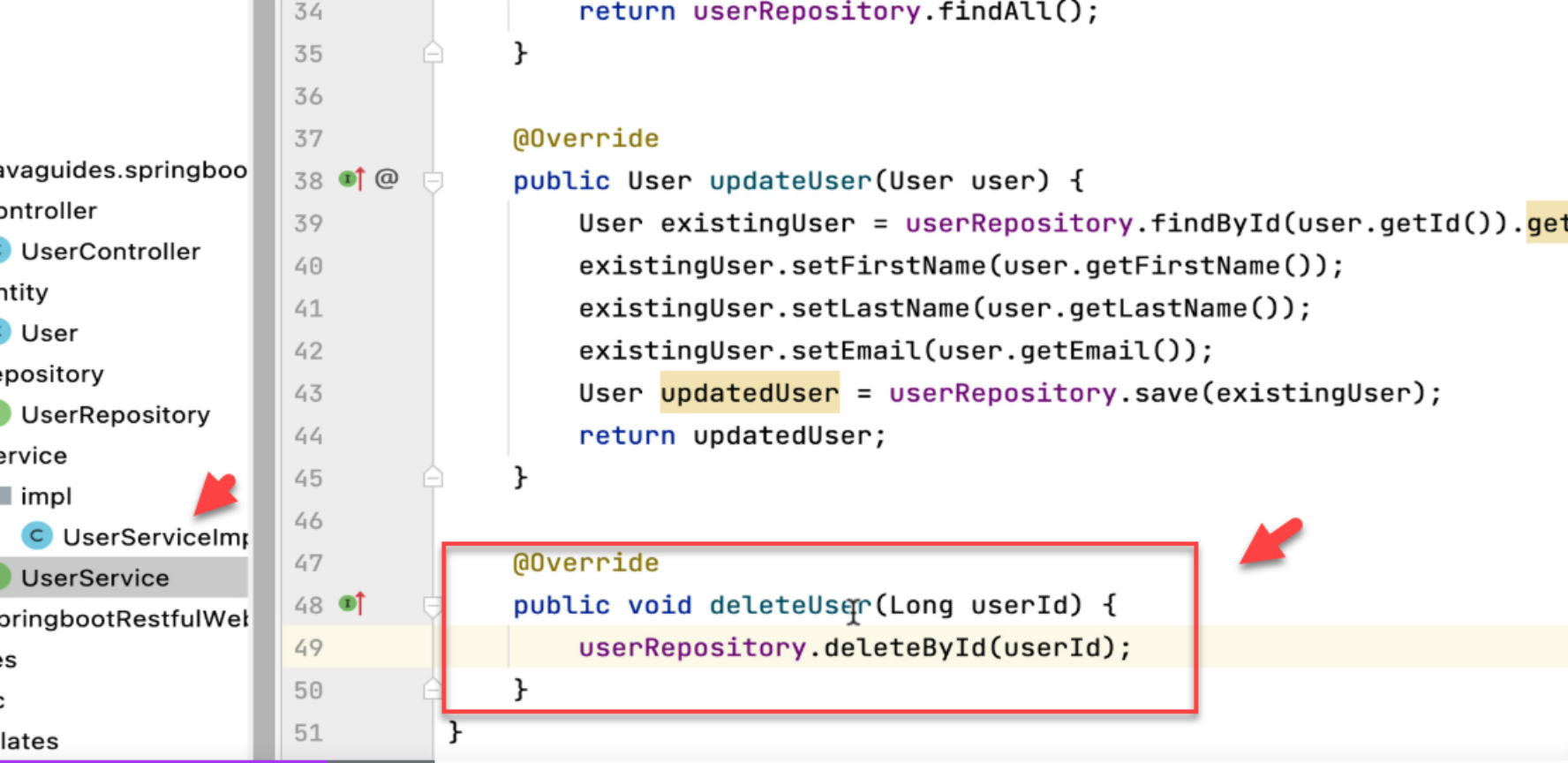

javaguides › springboot › service › impl › UserServiceImpl › updateUser
SpringbootRestfulWebservicesApplication
UserService.java
UserServiceImpl.java
Objects.java
Strings.class
String.java
UserController.java
27 public User getUserById(Long userId) {
28 Optional<User> optionalUser = userRepository.findById(userId);
29 return optionalUser.get();
30 }
32 @Override
33 public List<User> getAllUsers() {
34 return userRepository.findAll();
35 }
37 @Override
38 public User updateUser(User user) {
39 User existingUser = userRepository.findById(user.getId()).get();
40 existingUser.setFirstName(user.getFirstName());
41 existingUser.setLastName(user.getLastName());
42 existingUser.setEmail(user.getEmail());
43 User updatedUser = userRepository.save(existingUser);
44 return updatedUser;
45 }
46 }
2

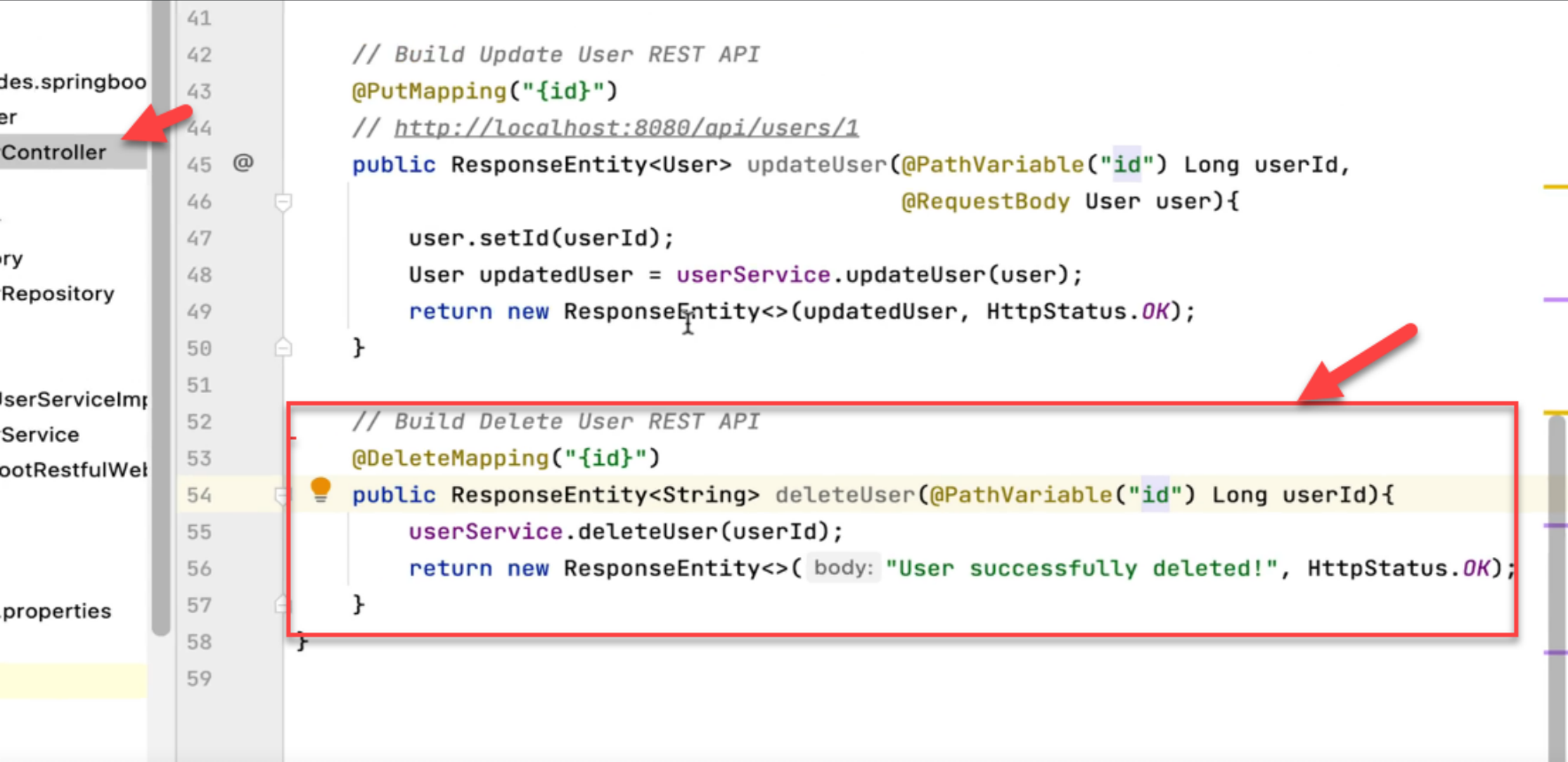

Local instance 3306
Administration
Schemas
SCHEMAS
Filter objects
myblog
myblogapp
registration_db
search_api
sms
sys
transaction_demo
user_management
Tables
users
Views
Stored Procedures
Functions
Object Info
Session
Table: users
Columns:
id bigint AI PK
email varchar(255)
first_name varchar(255)
last_name varchar(255)
Query 1
users
users
users
users
users
Limit to 1000 rows
1 • SELECT * FROM user_management.users;
100%
1:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
id email first_name last_name
1 ramesh@gmail.com Ramesh Fadatare
2 tom@gmail.com tom cruise
3 umesh@gmail.com umesh fadatare
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
users 1
Apply
Revert
Action Output
Time Action Response Duration / Fetch Time
1 13:04:31 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00067 sec / 0.0000...
2 13:05:06 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00038 sec / 0.000...

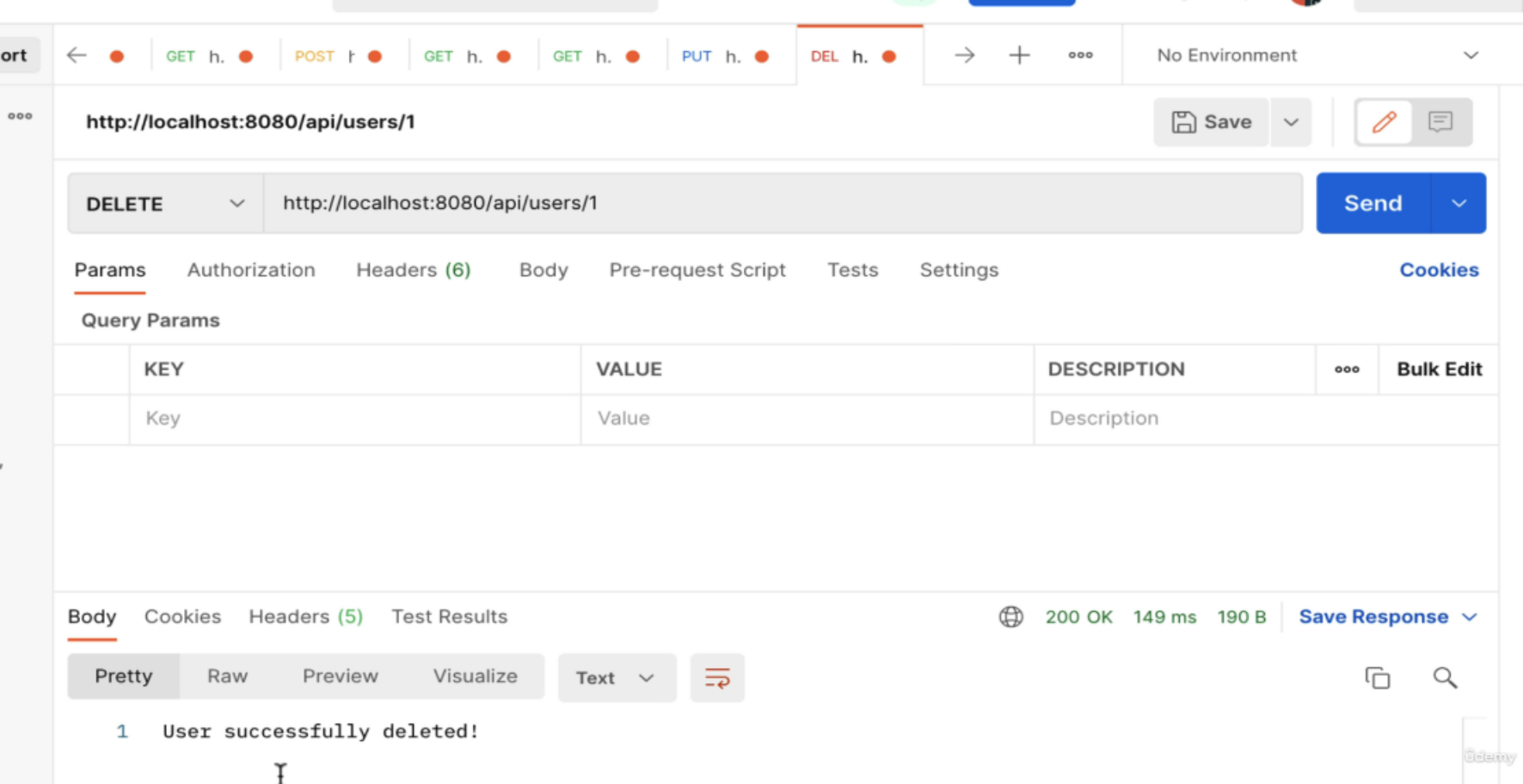

Postman
Home
Workspaces
Reports
Explore
Search Postman
Invite
Upgrade
Spring Boot Unit Testing
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
http://localhost:8080/api/users/1
1
Save
2
PUT
http://localhost:8080/api/users/1
4
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstName": "ram",
"lastName": "jadhav",
"email":"ram@gmail.com"
}
3
Body
Cookies
Headers (5)
Test Results
200 OK
317 ms
234 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"firstName": "ram",
"lastName": "jadhav",
"email": "ram@gmail.com"
}
5

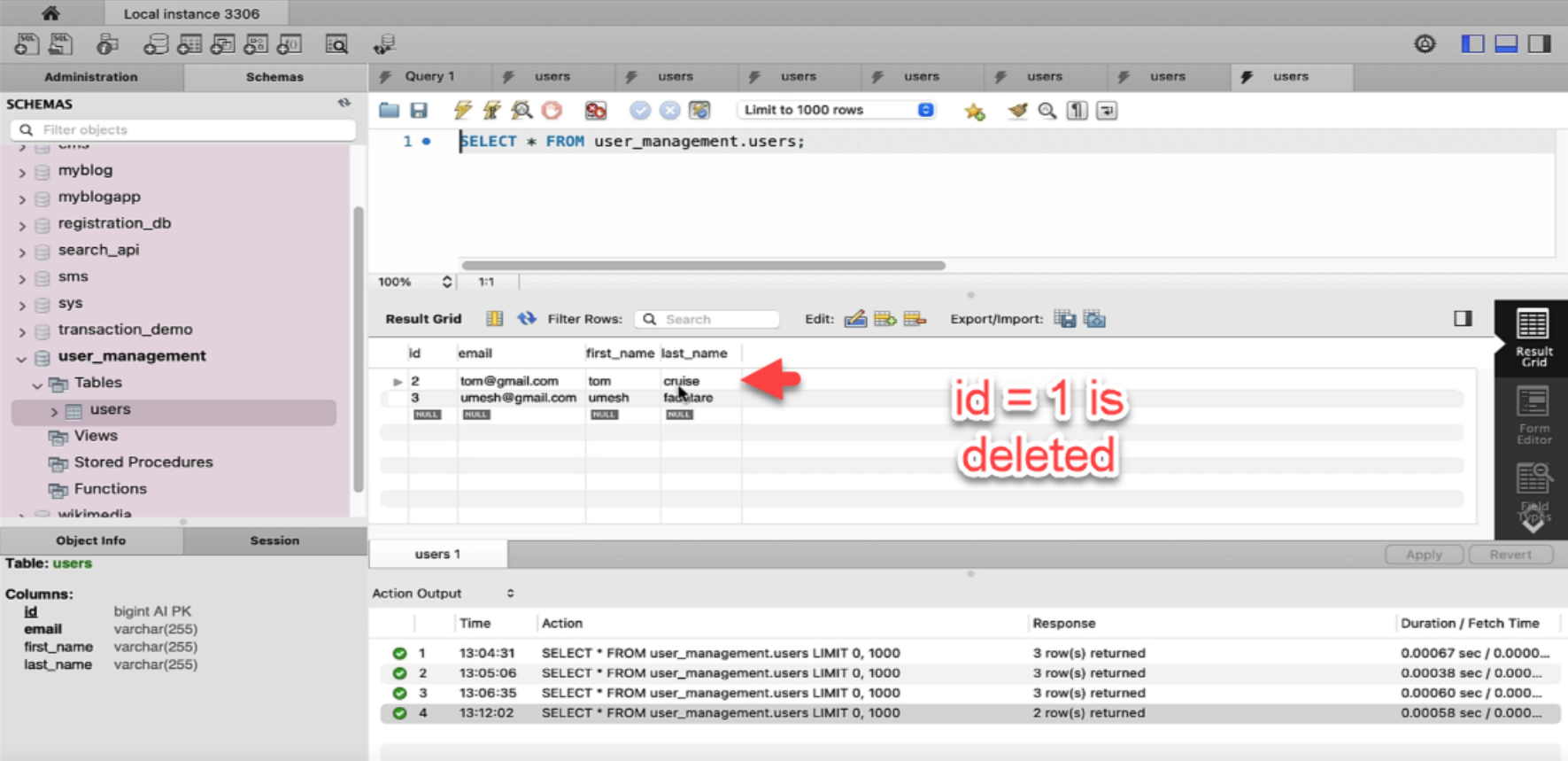

Local instance 3306
Administration
Schemas
SCHEMAS
Filter objects
myblog
myblogapp
registration_db
search_api
sms
sys
transaction_demo
user_management
Tables
users
Views
Stored Procedures
Functions
wikimedia
Object Info
Session
Table: users
Columns:
id bigint AI PK
email varchar(255)
first_name varchar(255)
last_name varchar(255)
Query 1
users
users
users
users
users
users
Limit to 1000 rows
1 • SELECT * FROM user_management.users;
100%
1:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
id email first_name last_name
1 ram@gmail.com ram jadhav
2 tom@gmail.com tom cruise
3 umesh@gmail.com umesh fadatare
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
users 1
Apply
Revert
Action Output
Time Action Response Duration / Fetch Time
1 13:04:31 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00067 sec / 0.0000...
2 13:05:06 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00038 sec / 0.000...
3 13:06:35 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00060 sec / 0.000...


avaguides.springboo
ontroller
UserController
ntity
User
epository
UserRepository
ervice
impl
UserServiceImp
UserService
pringbootRestfulWeb
es
lates
3 import net.javaguides.springboot.entity.User;
4
5 import java.util.List;
6
1 related problem
7 public interface UserService {
8 User createUser(User user);
9
10 User getUserById(Long userId);
11
12 List<User> getAllUsers();
13
14 User updateUser(User user);
15
16 void deleteUser(Long userId);
17 }
18
1


avaguides.springboo
ontroller
UserController
ntity
User
epository
UserRepository
ervice
impl
UserServiceImp
UserService
pringbootRestfulWeb
es
lates
34 return userRepository.findAll();
35 }
36
37 @Override
38 @ public User updateUser(User user) {
39 User existingUser = userRepository.findById(user.getId()).get
40 existingUser.setFirstName(user.getFirstName());
41 existingUser.setLastName(user.getLastName());
42 existingUser.setEmail(user.getEmail());
43 User updatedUser = userRepository.save(existingUser);
44 return updatedUser;
45 }
46
47 @Override
48 public void deleteUser(Long userId) {
49 userRepository.deleteById(userId);
50 }
51 }


des.springboo
er
Controller
ory
Repository
UserServiceImp
Service
ootRestfulWeb
properties
41
42 // Build Update User REST API
43 @PutMapping("{id}")
44 // http://localhost:8080/api/users/1
45 @ public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
46 @RequestBody User user){
47 user.setId(userId);
48 User updatedUser = userService.updateUser(user);
49 return new ResponseEntity<>(updatedUser, HttpStatus.OK);
50 }
51
52 // Build Delete User REST API
53 @DeleteMapping("{id}")
54 public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
55 userService.deleteUser(userId);
56 return new ResponseEntity<>(body: "User successfully deleted!", HttpStatus.OK);
57 }
58 }
59


GET h.
POST h
GET h.
GET h.
PUT h.
DEL h.
No Environment
http://localhost:8080/api/users/1
Save
DELETE
http://localhost:8080/api/users/1
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
200 OK
149 ms
190 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1 User successfully deleted!


Local instance 3306
Administration
Schemas
SCHEMAS
Filter objects
myblog
myblogapp
registration_db
search_api
sms
sys
transaction_demo
user_management
Tables
users
Views
Stored Procedures
Functions
Object Info
Session
Table: users
Columns:
id bigint AI PK
email varchar(255)
first_name varchar(255)
last_name varchar(255)
Query 1
users
Limit to 1000 rows
1 SELECT * FROM user_management.users;
100%
1:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
id email first_name last_name
2 tom@gmail.com tom cruise
3 umesh@gmail.com umesh fadetare
NULL NULL NULL NULL
id = 1 is deleted
Result Grid
Form Editor
Field Types
users 1
Apply
Revert
Action Output
Time Action Response Duration / Fetch Time
1 13:04:31 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00067 sec / 0.0000...
2 13:05:06 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00038 sec / 0.000...
3 13:06:35 SELECT * FROM user_management.users LIMIT 0, 1000 3 row(s) returned 0.00060 sec / 0.000...
4 13:12:02 SELECT * FROM user_management.users LIMIT 0, 1000 2 row(s) returned 0.00058 sec / 0.000...