

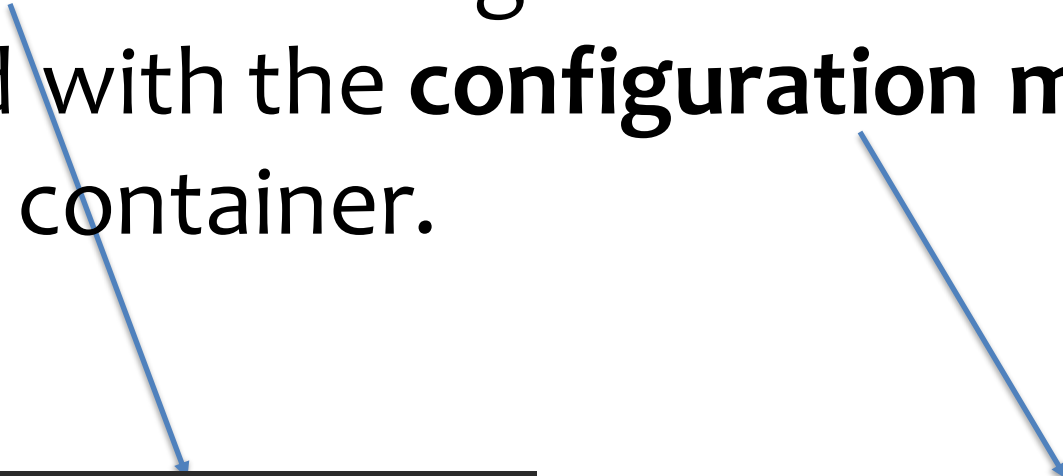
Session 2: Spring Core & Restful Web services

Learning goals

- Spring Core
 - IoC Container & Beans
 - Dependency Injection
- Restful Web services

Spring Core – IoC Container & Beans

- **A Spring IoC container** manages one or more beans. These beans are created with the **configuration metadata** that you supply to the container.



```
AccountService accountService = applicationContext
    .getBean(AccountService.class);
accountService.saveAccount(account);
```

```
no usages
@Configuration
public class BeanConfiguration {
    no usages
    @Bean
    AccountService accountService() {
        return new AccountServiceImpl();
    }
}
```

Spring Core – IoC Container & Beans

- In Spring, the objects that form **the backbone of your application** and that are managed by the Spring IoC *container* are called *beans*.



Spring Core – IoC Container & Beans

- Bean Scopes:

```
@Configuration
public class BeanConfiguration {
    no usages
    @Bean
    @Scope("singleton")
    AccountService accountService() {
        return new AccountServiceImpl();
    }
}
```

```
@Configuration
public class BeanConfiguration {
    no usages
    @Bean
    @Scope("prototype")
    AccountService accountService() {
        return new AccountServiceImpl();
    }
}
```

Spring Core – IoC Container & Beans

- We can also mark a Class to Bean by
 - @Component
 - @Controller
 - @RestController
 - @Service
 - @Repository

What is differences between them?



Spring Core – IoC Container & Beans

- @Controller

```
no usages
@Controller
@RequestMapping("/accounts")
public class AccountController {
```

Spring Core – IoC Container & Beans

- @RestController

```
no usages
@RestController
@RequestMapping("/accounts")
public class AccountController {
```


Spring Core – IoC Container & Beans

- @Service

no usages

@Service

```
public class AccountServiceImpl implements AccountService{
```

Spring Core – IoC Container & Beans

- @Repository

```
no usages
@Repository
public class AccountRepositoryImpl implements AccountRepository{
    no usages
    @Override
    public AccountRepository insert(Account account) {
        //handle code
        return null;
    }
}
```

Spring Core – IoC Container & Beans

- @Component

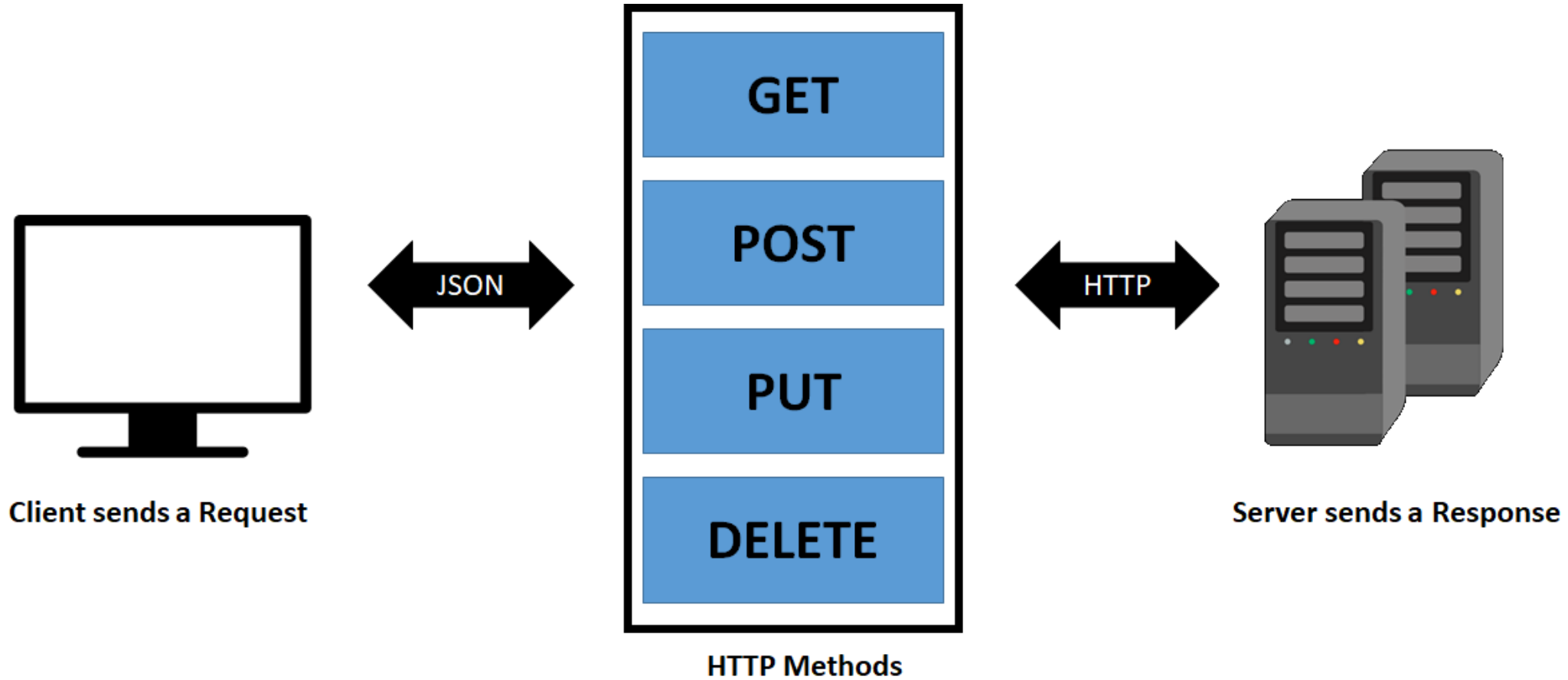
no usages

```
@Component|
```

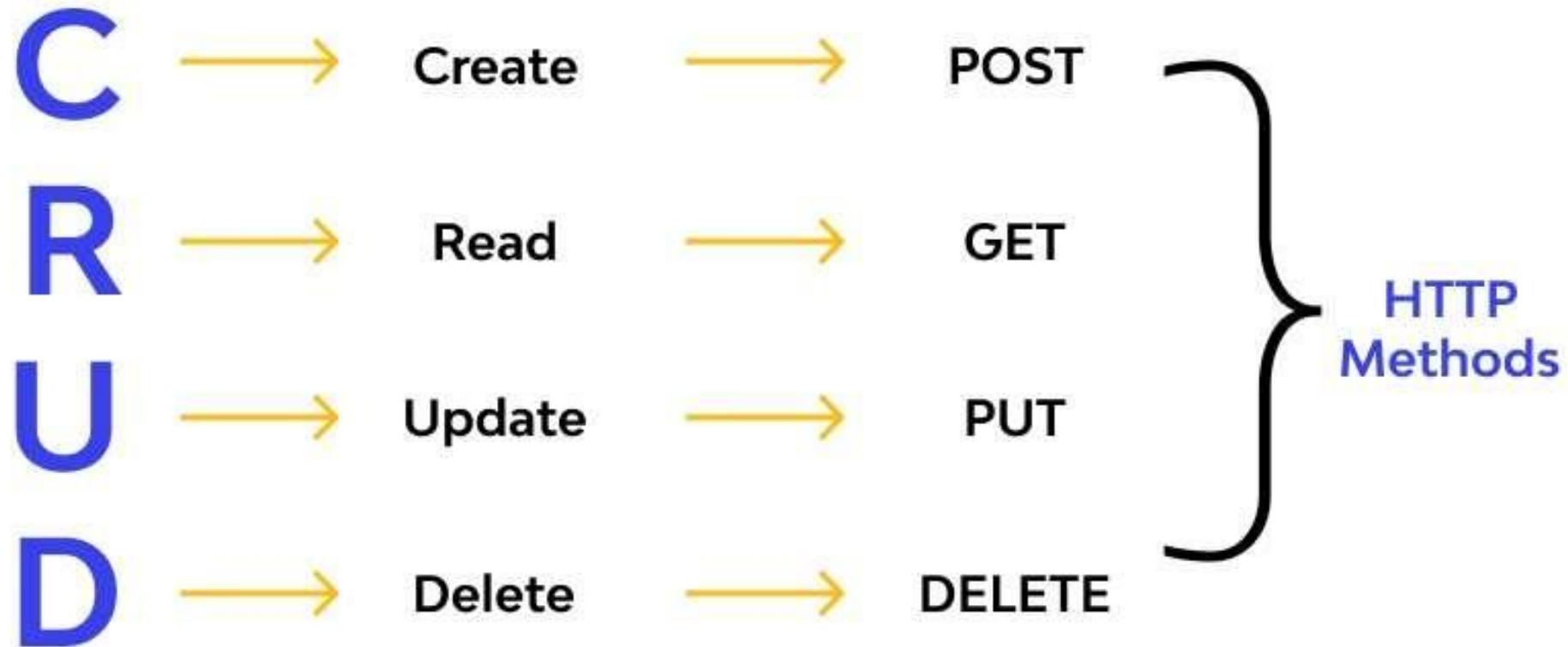
```
public class AccountFilterHelper {
```

- **There are three common ways of injecting dependencies:**
 - Property injection
 - Constructor injection
 - Method injection

Restful web services



CRUD



Example - CRUD APIs

Create Web APIs for CRUD operation

User



GET

/users Retrieve all users



POST

/users Create a new user



GET

/users/{id} Retrieve a user by ID



PUT

/users/{id} Update an existing user



DELETE

/users/{id} Delete a user by ID



Example - CRUD APIs

Create class **User**:

```
9 usages
public class User {
    no usages
    private Long id;
    no usages
    private String name;
    no usages
    private LocalDate dateOfBirth;
    no usages
    private String email;

    //generate getters and setters
}
```


Example - CRUD APIs

Create class **UserController**:

```
no usages
@RestController
@RequestMapping("/users")
public class UserController {
```

Example - CRUD APIs

Create class **Map** of user:

8 usages

```
private static final Map<Long, User> userMap = new HashMap<>();  
static {  
    userMap.put(1L, new User( id: 1L, name: "Nguyen Van A", LocalDate.of( year: 2000, month: 12, dayOfMonth: 12), email: "a@gmail.com"));  
}
```

Example - CRUD APIs

Get a list of users:

```
no usages  
  
@GetMapping   
public List<User> getAll() {  
    //Handle code  
}
```

Example - CRUD APIs

Get a specific user from a list:

no usages

```
@GetMapping(🌐↓("/{id}"))
```

```
public User getById(@PathVariable("<u>id</u>") Long id) {
```

```
    //Handle code
```

```
}
```

Example - CRUD APIs

Create a user:

```
no usages
@PostMapping
public User create(@RequestBody User user) {
    //Handle code
}
```

Example - CRUD APIs

Update an existing user:

no usages

@PutMapping 

```
public User update(@RequestBody User user) {
```

```
    //Handle code
```

```
}
```

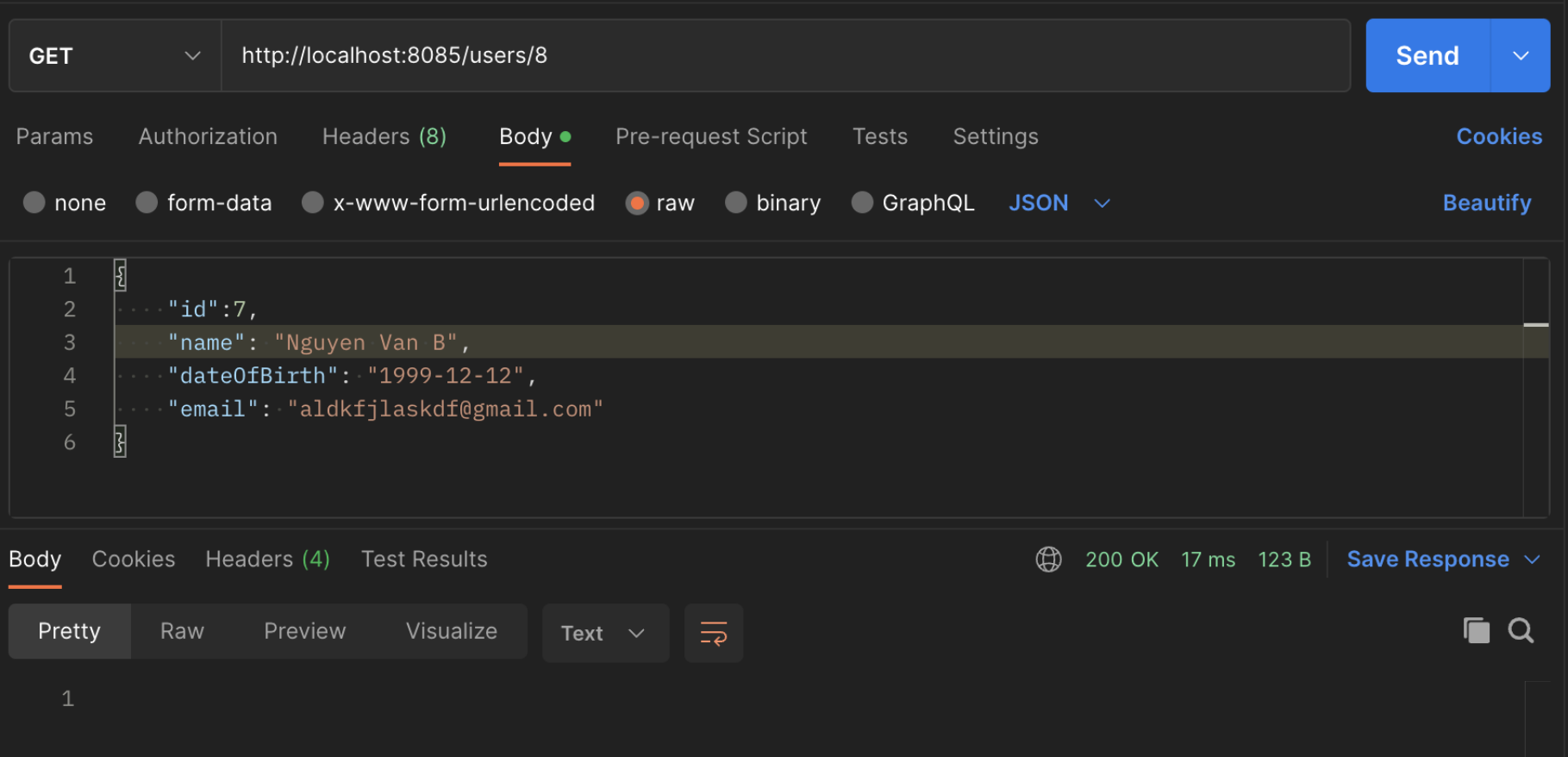
Example - CRUD APIs

Delete a specific user:

```
no usages
@DeleteMapping(🌐↓"/{id}")
public void deleteById(@PathVariable("id") Long id) {
    //Handle code
}
```

Example - CRUD APIs

Get a specific user not found?



The screenshot shows a REST client interface with a GET request to `http://localhost:8085/users/8`. The request body is a JSON object representing a user with ID 7. The response status is 200 OK.

Request:

- Method: GET
- URL: `http://localhost:8085/users/8`
- Body (JSON):

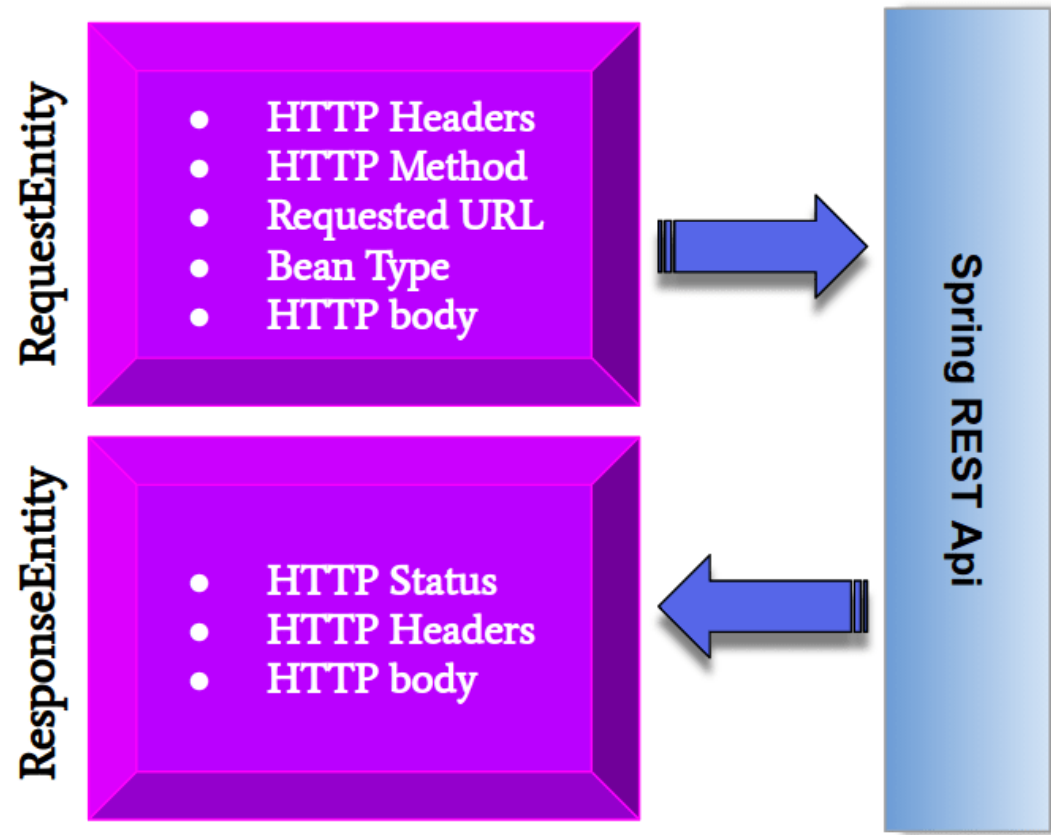
```
{  1  {  2    "id": 7,  3    "name": "Nguyen Van B",  4    "dateOfBirth": "1999-12-12",  5    "email": "aldkfjlaskdf@gmail.com"  6  }
```

Response:

- Status: 200 OK
- Time: 17 ms
- Size: 123 B

Example - CRUD APIs

Custom Response with **ResponseEntity**:



Custom Response with **ResponseEntity**:

Constructor Summary

Constructors

Constructor	Description
<code>ResponseEntity(HttpStatusCode status)</code>	Create a <code>ResponseEntity</code> with a status code only.
<code>ResponseEntity(MultiValueMap<String ,String > headers, HttpStatusCode status)</code>	Create a <code>ResponseEntity</code> with headers and a status code.
<code>ResponseEntity(T body, HttpStatusCode status)</code>	Create a <code>ResponseEntity</code> with a body and status code.
<code>ResponseEntity(T body, MultiValueMap<String ,String > headers, int rawStatus)</code>	Create a <code>ResponseEntity</code> with a body, headers, and a raw status code.
<code>ResponseEntity(T body, MultiValueMap<String ,String > headers, HttpStatusCode status)</code>	Create a <code>ResponseEntity</code> with a body, headers, and a status code.

Example - CRUD APIs

Get a specific user from a list:

no usages

```
@GetMapping("/{id}")  
public ResponseEntity<User> getById(@PathVariable("id") Long id) {  
    if (!userMap.containsKey(id)) {  
        //Handle code  
    } else {  
        //Handle code  
    }  
}
```

Example - CRUD APIs


Get a list of users:

```
@GetMapping  
public ResponseEntity<List<User>> getAll() {  
    if (//condition to check if there is not any user in a userMap) {  
        //handle code  
    } else {  
        //handle code  
    }  
}
```

Example - CRUD APIs

Create a user:

no usages

`@PostMapping` 

```
public User create(@RequestBody User user) {  
    if (//condition to check if user is existed from list) {  
        //handle code  
    } else {  
        //handle code  
    }  
}
```

Example - CRUD APIs

Update an existing user:

no usages

@PutMapping 

```
public User update(@RequestBody User user) {  
    if (//condition to check if user is existed in userMap) {  
        //handle code  
    } else {  
        //handle code  
    }  
}
```

Example - CRUD APIs

Delete a specific user:

no usages

```
@DeleteMapping("/{id}")
public void deleteById(@PathVariable("id") Long id) {
    if (//condition to check if user is existed in userMap) {
        //handle code
    } else {
        //handle code
    }
}
```

*Keeping up those **inspiration** and the **enthusiasm** in the **learning path**.
Let confidence to bring it into **your career path** for getting gain the **success**
as your expectation.*

Thank you

Contact

- Name: R2S Academy
- Email: daotao@r2s.edu.vn
- Hotline/Zalo: 0919 365 363
- Website: <https://r2s.edu.vn>
- Fanpage: <https://www.facebook.com/r2s.tuyendung>

Questions and Answers