Spring Boot Routing: A Complete Guide

1. Basic Routing Annotations

@RequestMapping

The most fundamental routing annotation in Spring Boot

```
@RestController
@RequestMapping("/api")
public class UserController {
    // Base path for all methods will be /api

    @RequestMapping(value = "/users", method = RequestMethod.GET)
    public List<User> getAllUsers() {
        // Handles GET /api/users
    }

    @RequestMapping(value = "/users/{id}", method = RequestMethod.GET)
    public User getUserById(@PathVariable Long id) {
        // Handles GET /api/users/1
    }
}
```

Simplified Routing Annotations

Spring Boot provides more specific annotations for different HTTP methods:

```
@RestController
@RequestMapping("/api/users")
public class UserController {
    // GET Request
    @GetMapping
    public List<User> getAllUsers() {
        // Handles GET /api/users
    }

    // GET Request with Path Variable
    @GetMapping("/{id}")
    public User getUserById(@PathVariable Long id) {
        // Handles GET /api/users/1
    }
    // POST Request
    @PostMapping
    public User createUser(@RequestBody User user) {
        // Handles POST /api/users
    }

    // PUT Request
    @PutMapping("/{id}")
    public User updateUser(@PathVariable Long id, @RequestBody User user) {
        // Handles PUT /api/users/1
    }

    // DELETE Request
    @DeleteMapping("/{id}")
    public void deleteUser(@PathVariable Long id) {
        // Handles DELETE /api/users/1
    }
}
```

2. Advanced Routing Techniques

Multiple Path Mappings

```
@GetMapping(value = {"/users", "/members", "/people"})
public List<User> listUsers() {
    // Handles GET /users, /members, and /people
}
```

Conditional Routing

java
Copy
```
@GetMapping(value = "/users", params = "type=admin")
public List<User> getAdminUsers() {
    // Only matches when type parameter is 'admin'
    // Example: /users?type=admin
}
```

Complex Path Matching

```
@GetMapping("/users/{category:[a-z]+}/{id:\\d+}")
public User getUserWithConstraints(
    @PathVariable String category,
    @PathVariable Long id
) {
    // Matches only when category contains lowercase letters
    // and id contains only digits
    // Example: /users/active/123
}
```

3. Request Parameter Handling

Query Parameters

```
@GetMapping("/search")
public List<User> searchUsers(
    @RequestParam(required = false) String name,
    @RequestParam(defaultValue = "10") int limit
) {
    // Handles /search?name=john&limit=20
}
```

Optional Parameters

```
@GetMapping("/users")
public List<User> filterUsers(
    @RequestParam(required = false) String department,
    @RequestParam(required = false) Double salary
) {
    // Handles /users?department=IT
    // or /users?salary=5000
}
```

4. Path Variable Techniques

Multiple Path Variables

```
@GetMapping("/users/{department}/{role}")
public List<User> getUsersByDepartmentAndRole(
    @PathVariable String department,
    @PathVariable String role
) {
    // Handles /users/engineering/manager
}
```

Optional Path Variables

```
@GetMapping({"/users", "/users/{id}"})
public ResponseEntity<?> getUsers(
    @PathVariable(required = false) Long id
) {
    if (id != null) {
        // Return specific user
    } else {
        // Return all users
    }
}
```

5. Routing with Regular Expressions

Regex Path Matching

```
@GetMapping("/users/{id:[0-9]+}")
public User getUserWithNumericId(@PathVariable Long id) {
    // Only matches numeric IDs
}
```

6. Cross-Origin Routing

CORS Configuration

```
@RestController
@CrossOrigin(origins = "http://example.com")
public class UserController {
    // Allows cross-origin requests from example.com
}

// Global CORS Configuration
@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/api/**")
                .allowedOrigins("http://example.com")
                .allowedMethods("GET", "POST", "PUT", "DELETE");
    }
}
```

7. Sub-Resource Routing

Nested Resources

```
@RestController
```

```java
@RequestMapping("/users/{userId}/orders")
public class OrderController {
    @GetMapping
    public List<Order> getUserOrders(@PathVariable Long userId) {
        // Handles GET /users/1/orders
    }

    @PostMapping
    public Order createUserOrder(
        @PathVariable Long userId,
        @RequestBody Order order
    ) {
        // Handles POST /users/1/orders
    }
}
```

8. Error Handling in Routing

Custom Error Handling

```java
@ControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(ResourceNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ErrorResponse handleResourceNotFound(ResourceNotFoundException ex) {
        return new ErrorResponse(ex.getMessage());
    }
}
```