Building REST APIs with Spring Boot

3.2 Creating REST Endpoints

Learning objectives

- Introduction to REST Controllers
- Understanding @RestController and @RequestMapping
- Creating simple GET endpoints
- Handling path and query parameters
- Returning JSON responses

What Is a REST Controller?

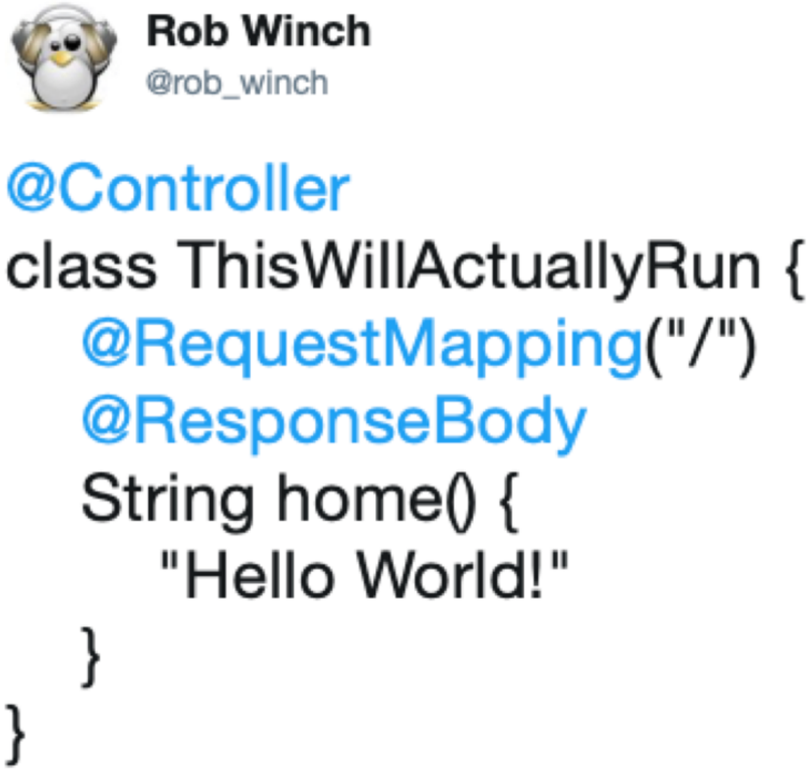
- Spring component that handles HTTP requests and responses
- Defined using the @RestController annotation
- Combines @Controller + @ResponseBody
- Automatically converts Java objects to JSON via Jackson
- Forms the backbone of RESTful APIs in Spring Boot

Learning objectives

Annotation	Purpose
@RestController	Marks class as REST API controller
@RequestMapping	Maps base URI path for all endpoints
@GetMapping, @PostMapping, etc.	Shortcut for specific HTTP methods
@PathVariable	Extracts variables from URL path
@RequestParam	Reads query parameters
@RequestBody	Binds request JSON body to Java object

Remember the tweet about Spring Boot?

- This is kind of what we will create now



Demo - Building a Simple REST API with Spring Boot

- Demonstrate key Spring Boot REST features
- Show how controllers expose endpoints
- Introduce JSON request/response handling
- Use simple Person objects as the domain
- Cover GET, POST, and PATCH
- Test endpoints using Postman or browser

What the Demo Will Show

- Returning JSON objects with @RestController
- Mapping HTTP methods with annotations
- Using @PathVariable for dynamic URLs
- Using @RequestParam for optional query parameters
- Creating new resources with POST
- Updating part of a resource with PATCH

Using @PathVariable and @RequestParam

- Path variables appear in the URL
- Query parameters come after “?”
- Path variables identify a specific resource
- Query parameters refine/filter data
- Browser can call both patterns
- Very common in real APIs

Lab – Building Your Own Task API

- Goal: Build a small REST API for managing simple tasks.
- You will create endpoints to:
  - Return a list of tasks
  - Return one task
  - Create a task with POST
  - Toggle task completion with PATCH
  - Work with JSON input and output

Lab Domain – Task

- Each task has:
  - id (int)
  - title (String)
  - completed (boolean)
- Use an in-memory list during this lab.

Lab – Required Endpoints

- 1. GET /api/tasks
  Return all tasks

- 2. GET /api/tasks/{id}
  Return a single task

- 3. POST /api/tasks
  Create a new task

  - JSON body: { "title": "Learn Spring" }

- 4. PATCH /api/tasks/{id}/toggle
  Flip completed from true → false or false → true

Lab Instructions

- 1. Create a Spring Boot project
- 2. Create a Task model class
- 3. Create a controller in /api/tasks
- 4. Maintain an in-memory list of tasks
- 5. Implement GET, POST, and PATCH endpoints
- 6. Start the application and test all endpoints
- 7. Verify JSON responses in your browser or Postman

Lab - Optional Challenge

- Add extra functionality:
- Support searching tasks by ?completed=true
- Add an auto-incrementing ID generator
- Prevent empty task titles

Lab Summary

- After completing this lab, you will understand:
  - Core controller mechanics
  - JSON input (POST) and partial updates (PATCH)
  - Managing simple state in memory
  - REST endpoint structure and naming conventions

Key Takeaways

- @RestController is the entry point for all REST APIs
- Annotations like @GetMapping, @PostMapping and @PatchMapping to map HTTP methods to Java methods.
- Annotations like @RequestParam and @PathVariable for input
- Spring Boot auto-converts Java objects into JSON
- You’ve just created your first working REST API!