CODED Mock Class

Creating a REST API with Spring Boot

Learning Objectives. You will…

- 1. Have basic knowledge about Spring Framework and Spring Boot (slides)
- 2. Understand the building blocks of a Spring Boot REST API (demo)
  - Spring Data JPA Repository to connect to the database
  - RestController class to handle HTTP requests
- 3. Be able to create a Spring Boot REST API (task)

Learning Objective 1

Basic knowledge about Spring Framework and Spring Boot

The Beginning of Spring

- Started out as code in a book by Rod Johnson
- A light weight alternative to Java EE and EJB
- Builds on Dependency Injection (Inversion of Control)
- Spring Framework 1.0 released in 2003

Rod Johnson

Rod Johnson is an enterprise Java architect specializing in scalable web applications. He has worked with both Java and J2EE since their release, and he is a member of JSR 154 Expert Group defining the Servlet 2.4 specification.

expert one-on-one

J2EE™ Design and Development

wrox

Updates, source code, and Wrox technical support at www.wrox.com

The Spring Framework - Benefits

- Very popular and powerful framework
- It provides a set of libraries and tools that simplify Java development
- Integrates with many popular open source frameworks
- Large number of modules providing different services

Spring Framework - Challenges

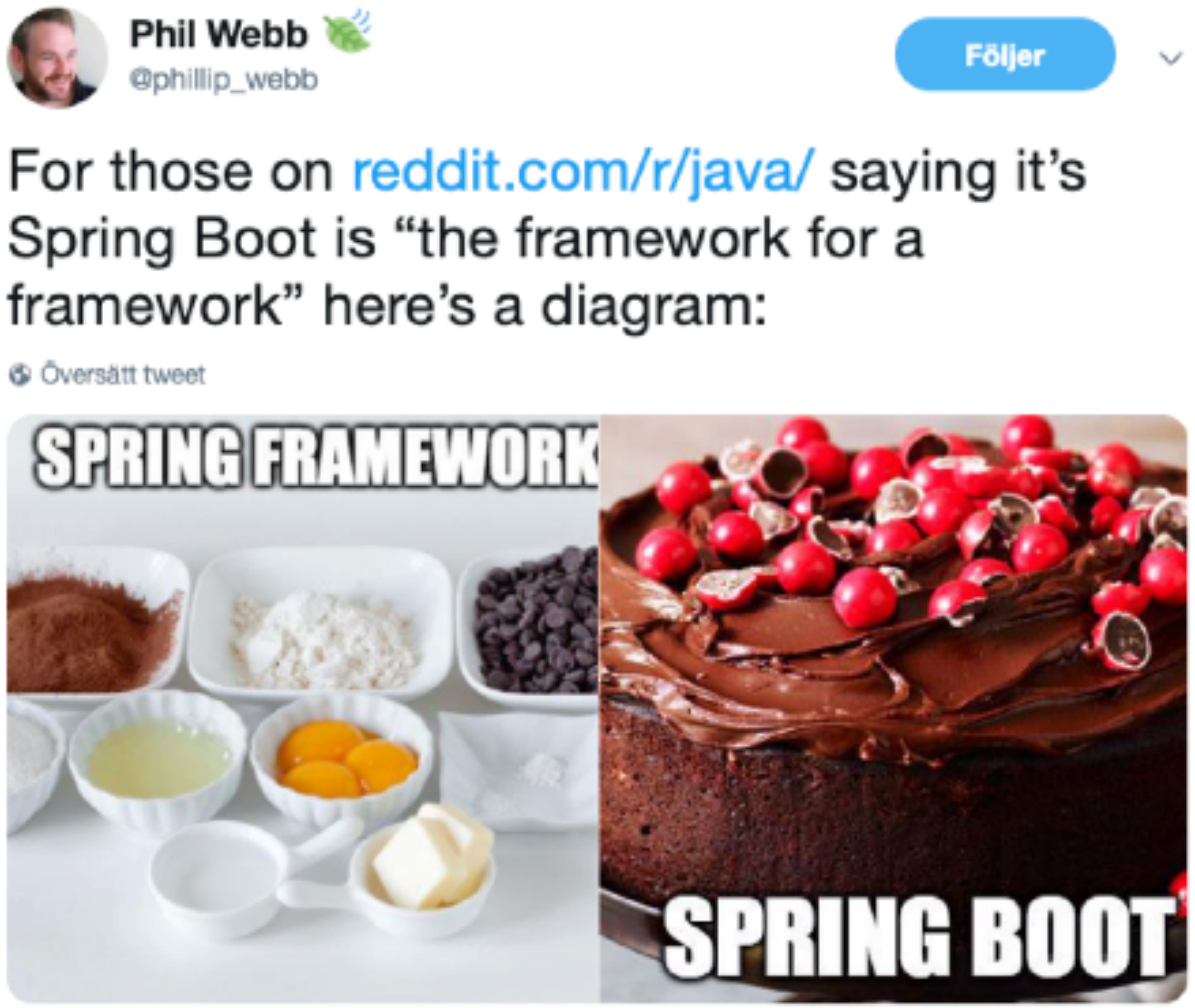
- The configuration can become complex
- Handling the versions of 3rd party dependencies can become complex
- With a large number of modules it's not as lightweight as in the beginning

Spring Boot

- Spring Boot tries to solve the problems with configuration and 3rd party libraries
- Spring Boot is an opinionated view of a Spring application
- Just tell Spring Boot what functionality you want (starter-dependencies)
- Enables Java developers to quickly start new projects
- The default configuration can easily be overridden by developers
- Production ready!

Spring Framework vs. Spring Boot

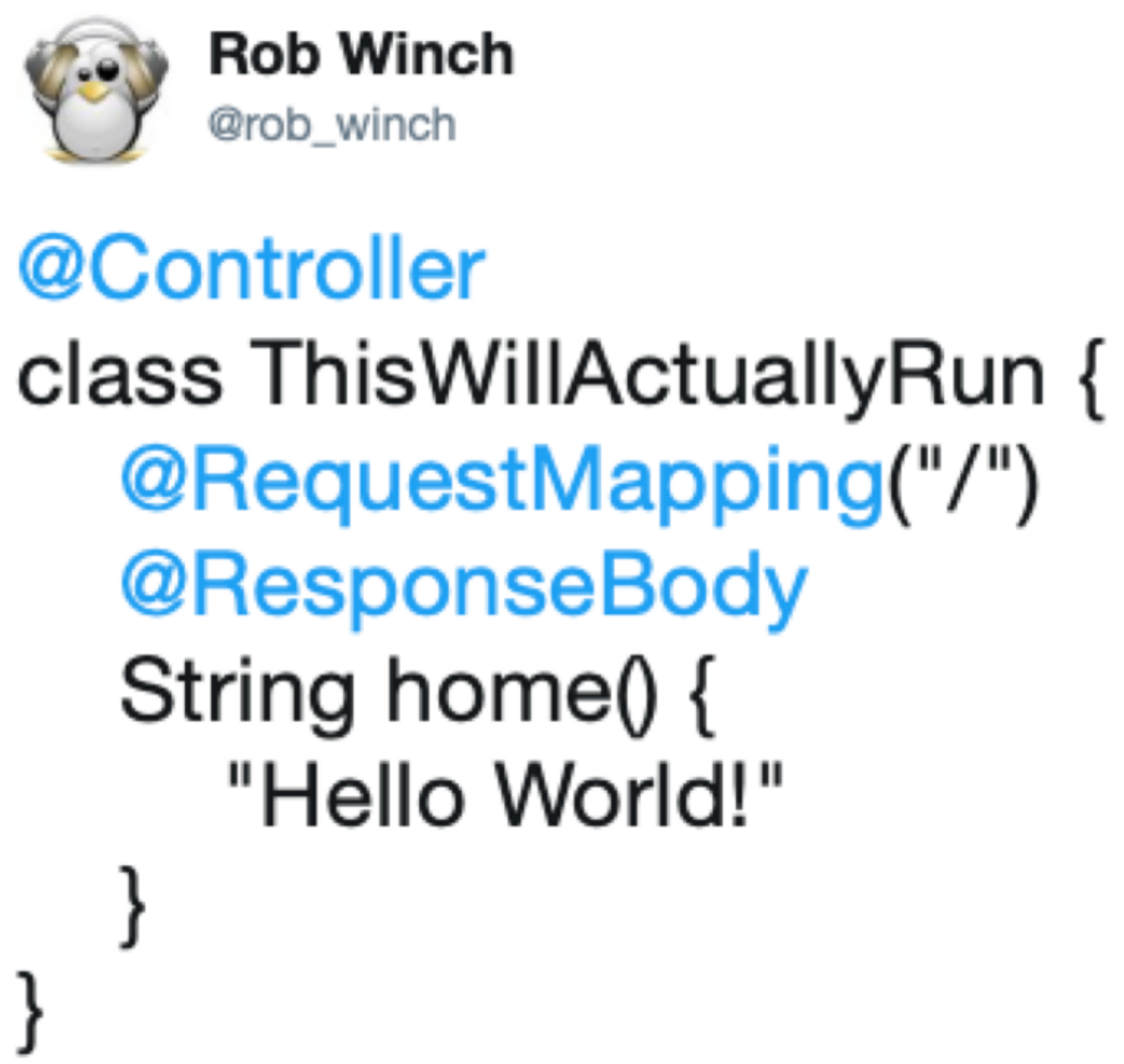
- Discuss - what do you think the image means?


Phil Webb
@phillip_webb

For those on reddit.com/r/java/ saying it's Spring Boot is "the framework for a framework" here's a diagram:

An entire Java web app in a tweet?

- But is this really Java?
- And will this really work by itself?
- Please discuss - what do you think?


Rob Winch
@rob_winch

```
@Controller
class ThisWillActuallyRun {
    @RequestMapping("/")
    @ResponseBody
    String home() {
        "Hello World!"
    }
}
```

The Magic of Spring Boot

- 1. Automatic configuration
  - Convention over configuration
  - Look at the dependencies
- 2. Starter dependencies
  - Pre-defined collections of dependencies
  - The versions are already tested and will work together

Learning Objective 2

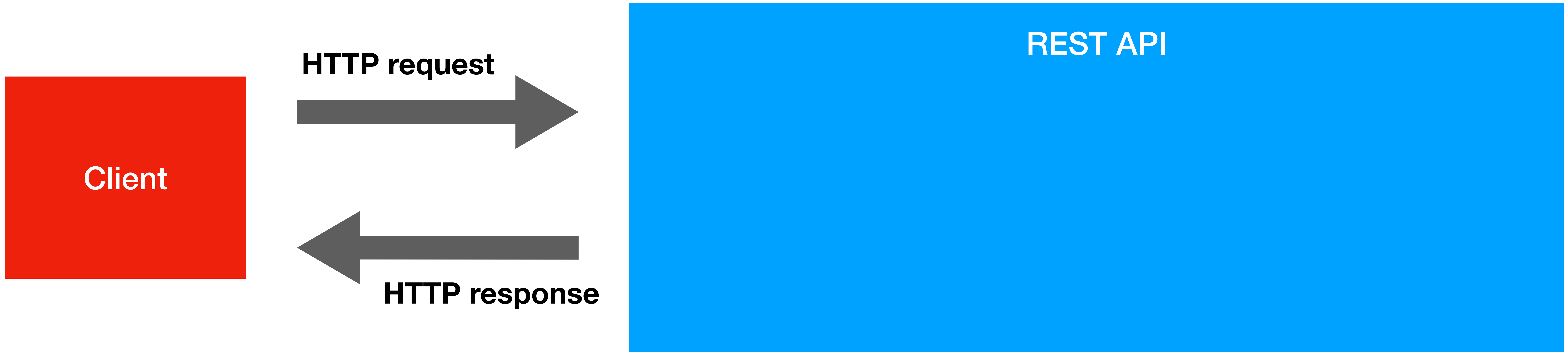
Basic building blocks for a Spring Boot REST API

What is a REST API?

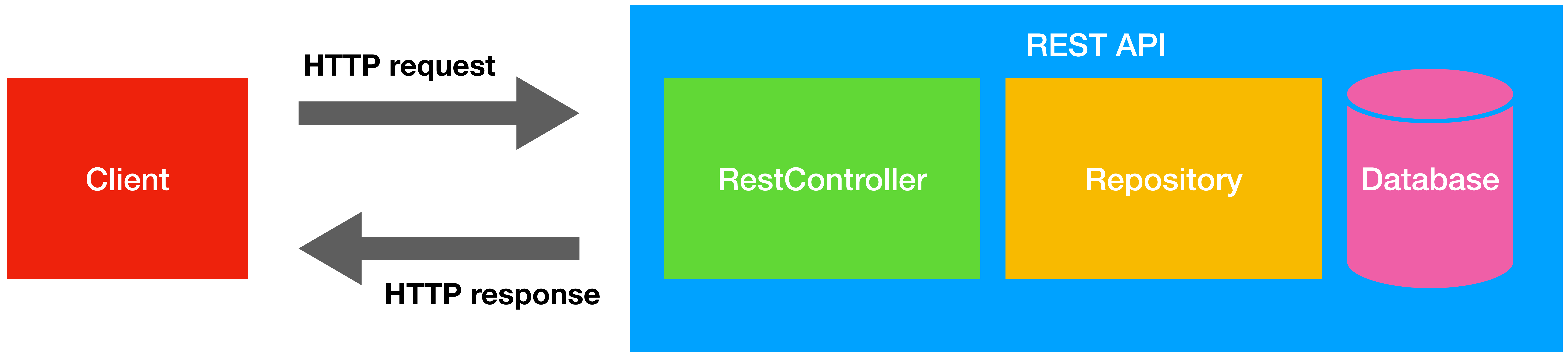
- REST stands for REpresentational State Transfer
- It's an architectural style for web services
- Often uses JSON as a data format (JavaScript Object Notation)
- Resources are identified with a URI - Uniform Resource Identifier
- Uses HTTP methods to specify the behavior
- Stateless - requests are independent from one another

Spring Boot REST API

- What kind of functionality do we need in the REST API - please discuss!



Spring Boot REST API

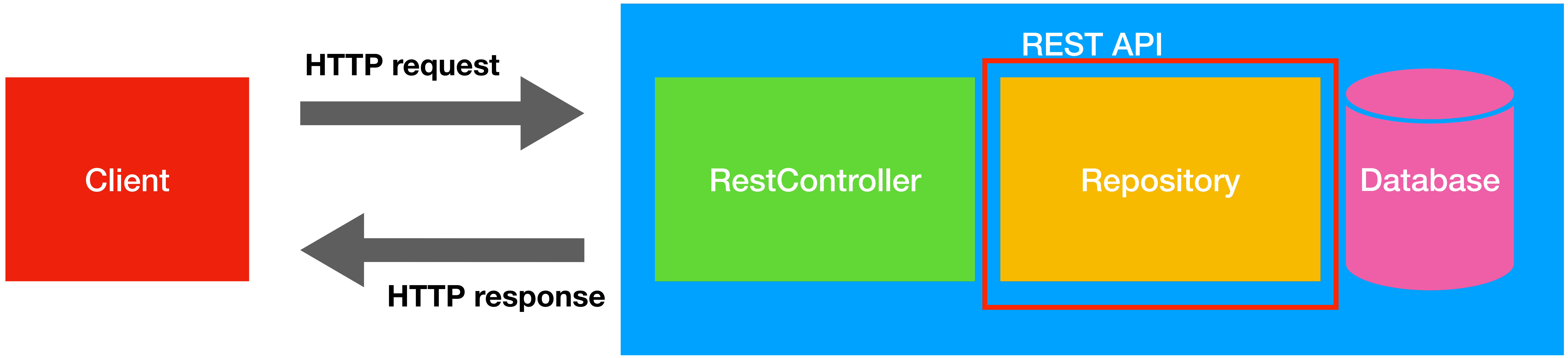


HTTP methods are like CRUD for databases

- POST - Create
- GET - Read
- PUT - Update
- DELETE - Delete

Demo 1 - The Repository

- Creating a Spring Data JPA Repository (by just creating an interface)
- Using JPA @Entity annotations
- Using this starter project: https://github.com/actleatraining/DemoProjectStarter

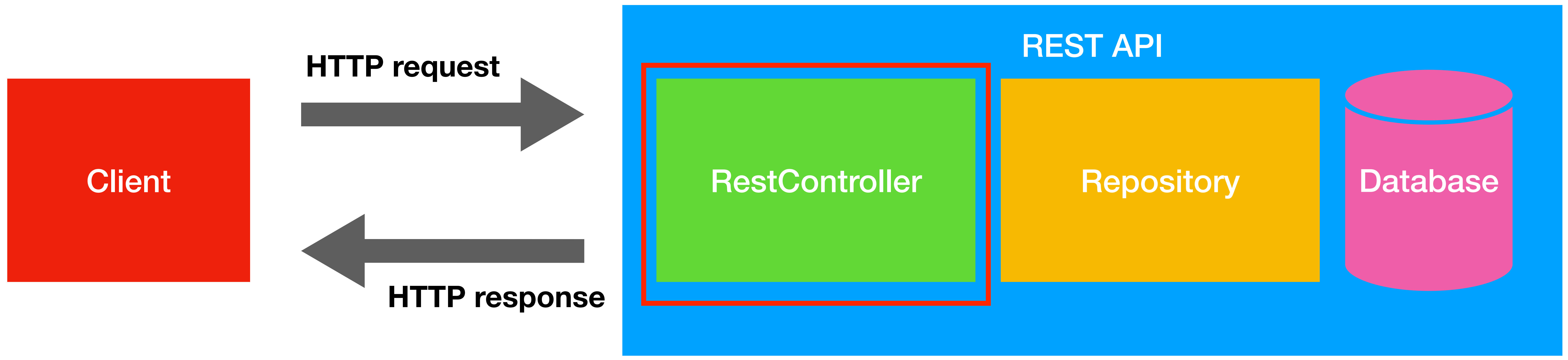


HTTP Requests in a REST API

HTTP method	URI	Description
GET	/book	Returns a list of all books
GET	/book/{id}	Returns one book with the requested id
POST	/book	Creates a new book (Book object as JSON in request body)
PUT	/book/{id}	Updates the book with the id (Book object as JSON in request body)
DELETE	/book/{id}	Deletes the book with the requested id

Demo 2 - The RestController

- Using @GetMapping, @PostMapping, @PutMapping, @DeleteMapping
- The URI is handled in the Mapping annotation
- Methods return objects or list of objects, automatically converted to JSON



1. Use the Mapping annotation

- For the HTTP method and the URI - Use @GetMapping, @PostMapping, @PutMapping, @DeleteMapping like this:

- @GetMapping("/book")

HTTP method	URI	Description
GET	/book	Returns a list of all books
GET	/book/{id}	Returns one book with the requested id
POST	/book	Creates a new book (Book object as JSON in request body)
PUT	/book/{id}	Updates the book with the id (Book object as JSON in request body)
DELETE	/book/{id}	Deletes the book with the requested id

2. Use a path variable for the id

- The {id} part of the URI after the second slash should be treated as a variable, like this as an input argument in the method:
- @PathVariable Long id

HTTP method	URI	Description
GET	/book	Returns a list of all books
GET	/book/$\boxed{\text{\{id\}}}$	Returns one book with the requested id
POST	/book	Creates a new book (Book object as JSON in request body)
PUT	/book/$\boxed{\text{\{id\}}}$	Updates the book with the id (Book object as JSON in request body)
DELETE	/book/$\boxed{\text{\{id\}}}$	Deletes the book with the requested id

3. Use @RequestBody for the data

- To get the data out for the request body, use the @RequestBody annotation on the input argument to the method:

- @RequestBody Book book

HTTP method	URI	Description
GET	/book	Returns a list of all books
GET	/book/{id}	Returns one book with the requested id
POST	/book	Creates a new book (Book object as JSON in request body)
PUT	/book/{id}	Updates the book with the id (Book object as JSON in request body)
DELETE	/book/{id}	Deletes the book with the requested id

Solution example for the demo

- https://github.com/actleatraining/DemoProjectSolution

Learning Objective 3

Create a Spring Boot REST API

The Task - Create a Spring Boot REST API

- The resource should be book like in the examples in the slides
- Use the TaskProjectStarter project (a database with books is prepared)
- Create a Book JPA Entity Class with variables, getters, setters and annotations
- Create a Spring Data JPA Repository Interface for Book
- Create a RestController class with the 5 REST API methods like in the examples
- Autowire the Repository into the RestController and use it from the methods
- Test the functionality with a web browser and Postman or similar tools
- Use the Starter Project: https://github.com/actleatraining/TaskProjectStarter

After the Task - examine the Solution example

- Examine and discuss the Solution Example:
- https://github.com/actleatraining/TaskProjectSolution

Learning Objectives. You will…

- 1. Have basic knowledge about Spring Framework and Spring Boot (slides)
- 2. Understand the building blocks of a Spring Boot REST API (demo)
  - Spring Data JPA Repository to connect to the database
  - RestController class to handle HTTP requests
- 3. Be able to create a Spring Boot REST API (task)