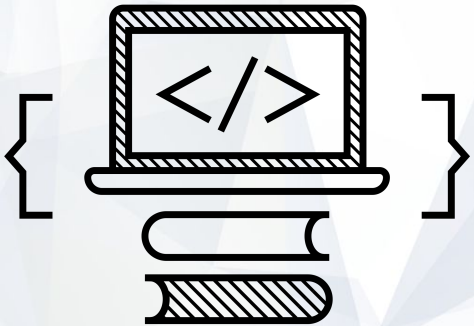
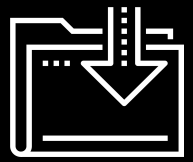


Intro to Spring Boot Part 2

Java Accelerator 7

Lesson 2.2



Learning Objectives

01 Use Spring testing libraries for TDD.

02 Use dependency injection in a Spring Boot application.

03 Build a REST web service using Spring Boot.

04 Design a simple REST API.

05 Exercise a REST API with Postman.

The Record Store API

The Record Store API

Retrieve all Records	• URI /records • HTTP method: GET • Request body: None • Response body: List of Records • Status code: 200 (OK)
Add a Record	• URI /records • HTTP method: POST • Request body: Record information • Response body: Record information with generated ID • Status code: 201 (Created)

The Record Store API

Retrieve a Record	• URI /records/{id} • HTTP method: GET • Request body: None • Response body: Record information • Status code: 200 (OK)
Update a Record	• URI /records/{id} • HTTP method: PUT • Request body: Record information • Response body: None • Status code: 204 (No Content)

The Record Store API

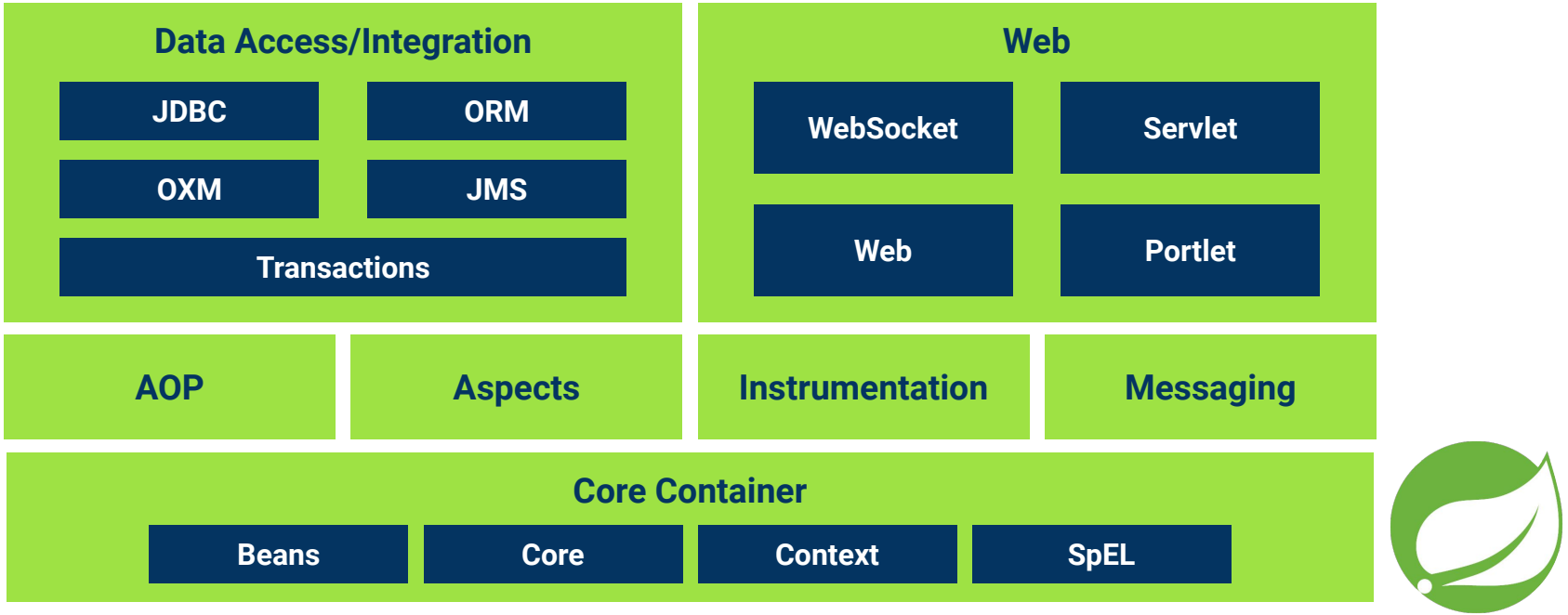
Delete a Record

- URI /records/{id}
- HTTP method: DELETE
- Request body: None
- Response body: None
- Status code: 204 (No Content)

This is the API that we’re going to build. First, as in all things TDD, we need to set up the tests for the controller. We'll use MockMvc for this.

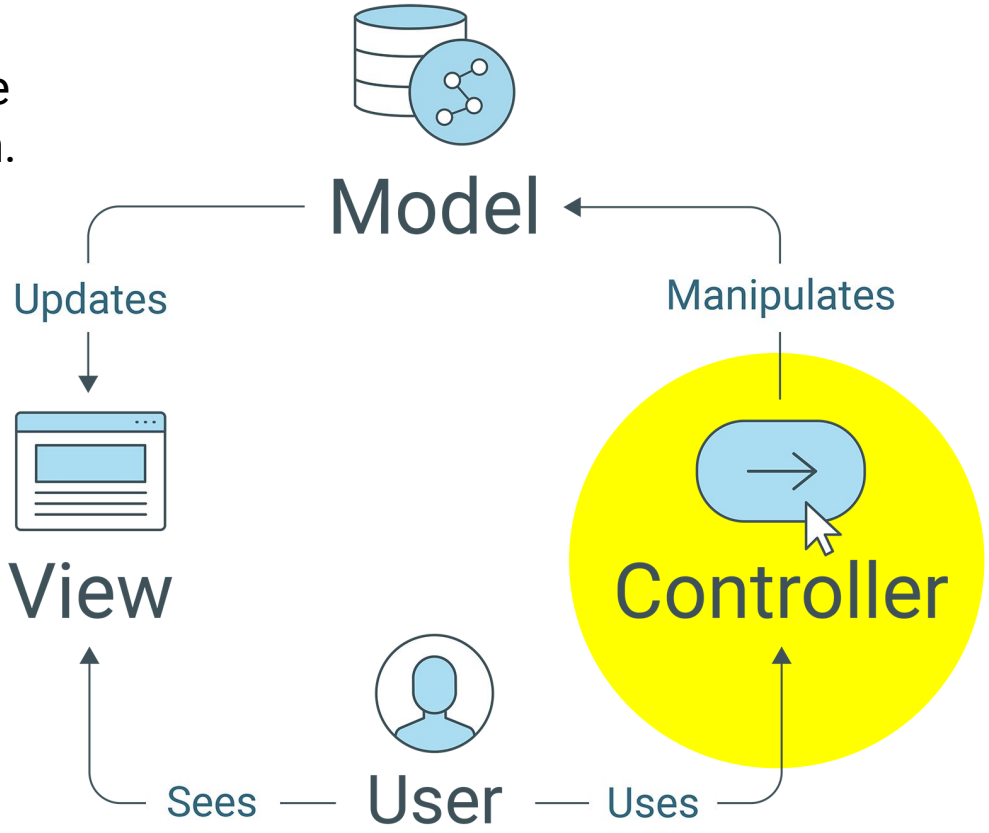
MockMvc

MockMvc is included in the Spring Framework dependencies—no additional dependencies required. We merely have to import it for use.



MockMvc

Using MockMvc to test controllers is similar to testing any component—we follow the Arrange-Act-Assert pattern.



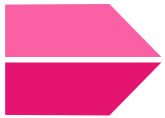
Time to Code

MockMVC Record Store

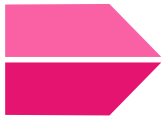
Suggested Time:

Spring Annotations

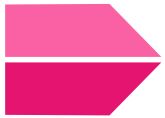
@RestController



This is a class-level annotation.



It marks the class as a REST controller and makes Spring aware of its existence.



This annotation configures Spring so that it treats all returned values from methods as JSON and sends those values back to the client.

@RequestMapping

This is a method-level annotation.

This maps an endpoint to a method that will handle requests to that endpoint.

It has two parameters:

01 **Value** contains the URI path for this mapping.

02 **Method** specifies the HTTP method for this mapping.

Must be unique across all controllers. In other words, you can only map a given endpoint to one method in an application.

@ResponseStatus

This is a method-level annotation.

This indicates the HTTP status code that is sent back when the method successfully handles the incoming request.

@PathVariable

This is a method-parameter annotation.

It maps a path variable (marked by `{}` in the path) to a method parameter.

The Spring Framework converts the value to the right type.

Place this annotation before the method parameter that you want to associate with the path variable.

If the name of the path variable is the same as the method variable, no further configuration is needed.

@RequestBody

This is a method-parameter annotation.

It maps JSON in the request body to a method parameter.

A component called the Jackson `ObjectMapper` takes care of converting JSON to Java and Java to JSON.

Time to Code

Echo Service

Suggested Time:

Time to Code

Record Store

Suggested Time:

Activity: Web Service Build

Suggested Time: