Intro to Spring Boot Part 1

Java Accelerator 7

Lesson 2.1

Purpose

01

Spring and Spring Boot have huge awareness in the enterprise Java development community.

02

Knowing how to build REST web services with Spring and Spring Boot is an essential professional skill.

03

Swagger is used by many organizations to help design and document their REST APIs.

Learning Objectives

01 Build a REST web service with Spring Boot.

02 Design a simple REST API.

Introduction to Spring

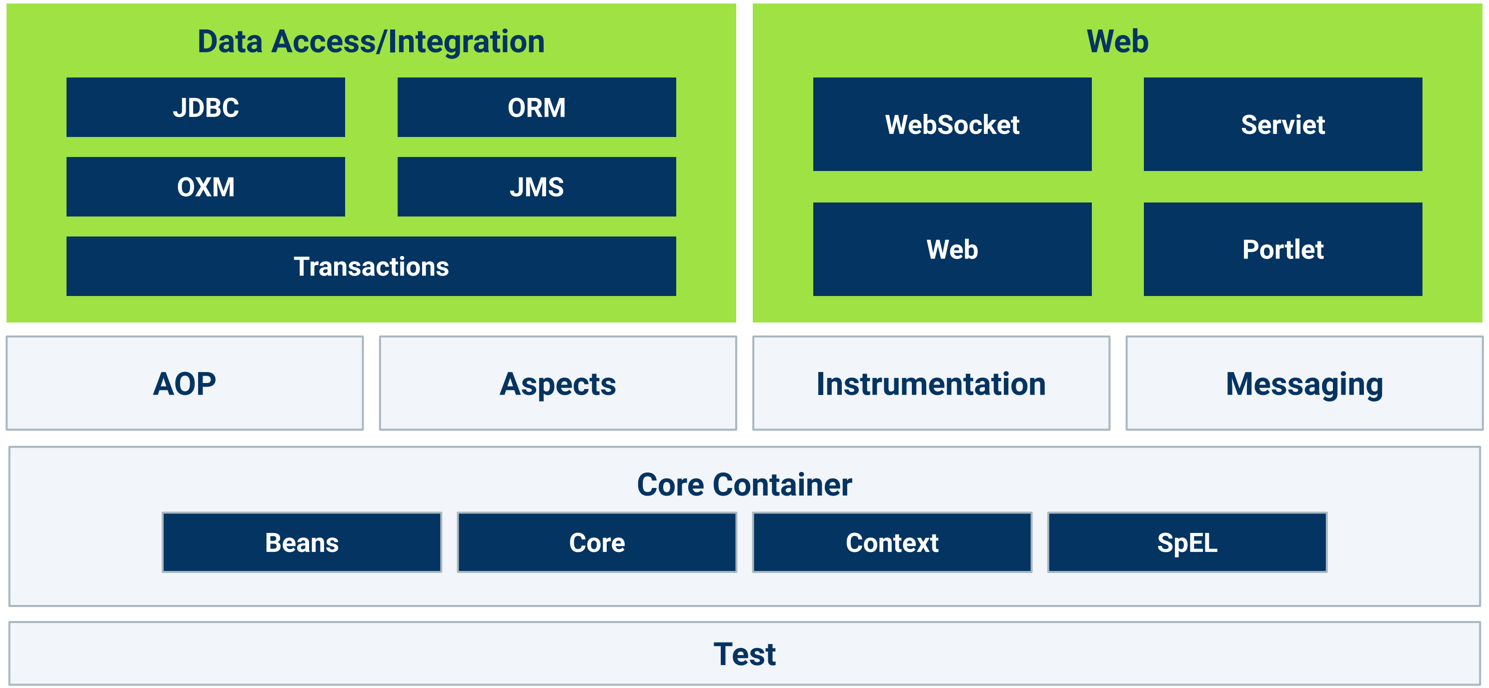
Spring is a collection of frameworks that make Java development easier and more lightweight.

Pro Tip:

In general, Spring gives us tools to create loosely coupled and easily configurable Java applications.

The Spring Project

The Spring Project has libraries and frameworks that can help with many types of application needs, including web development and data access.



How We'll Use Spring

Spring is a vast collection of libraries and frameworks.

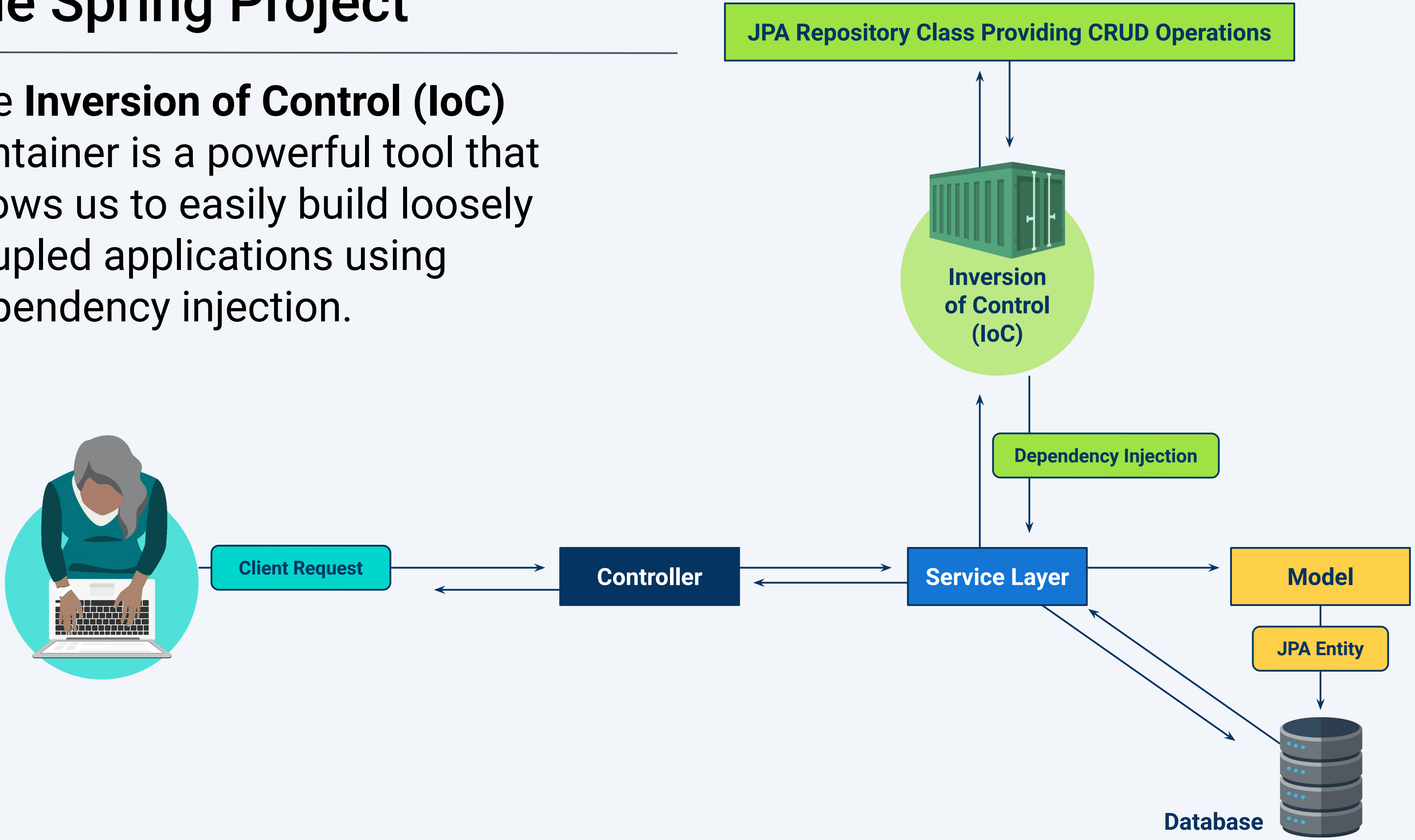
Through Spring Boot, we'll be using the following portions of the Spring ecosystem:

- Spring Framework
- Dependency Injection/IoC Container
- Spring MVC
- Testing
- Spring Data
- Spring Security

Spring Framework

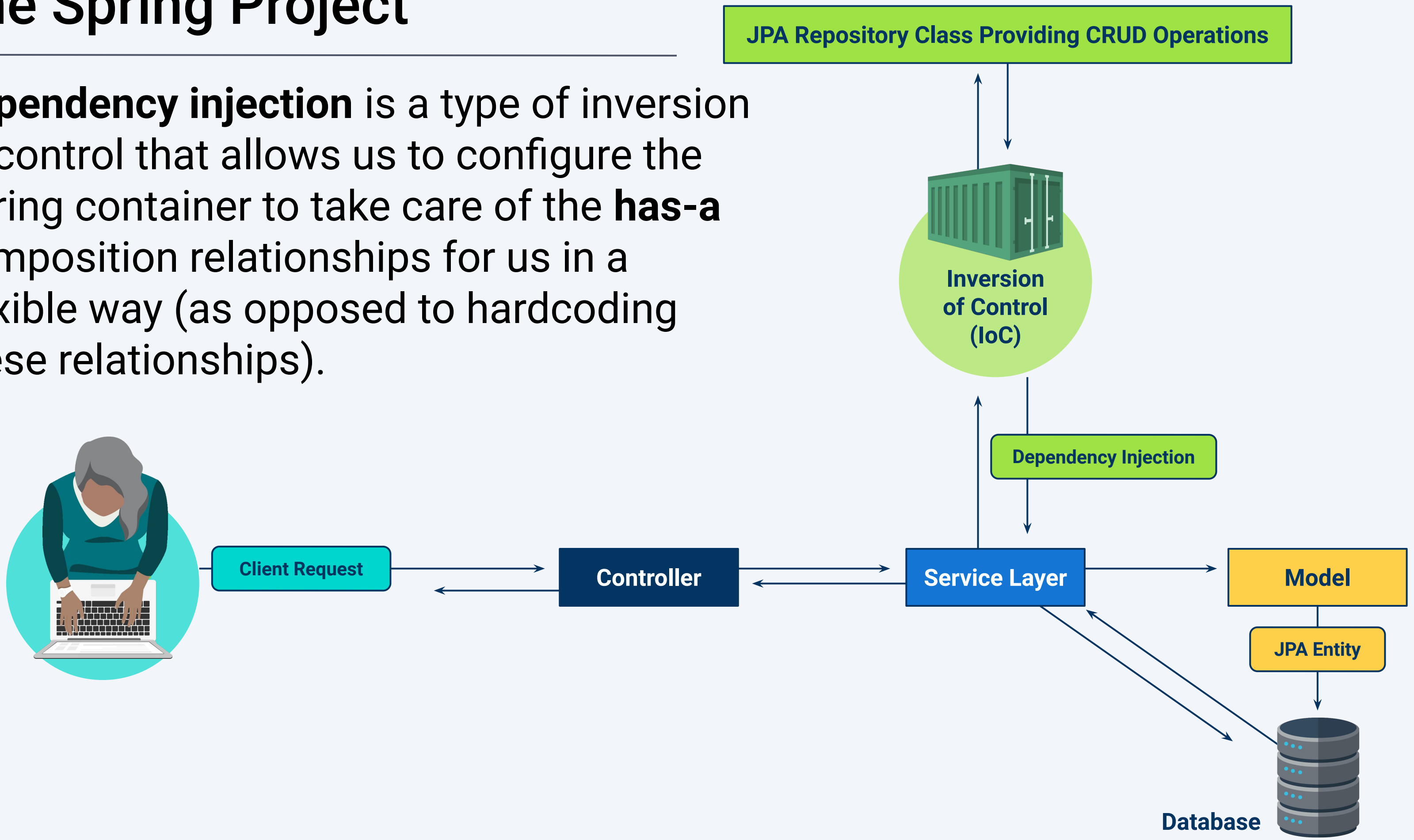
The Spring Project

The **Inversion of Control (IoC)** container is a powerful tool that allows us to easily build loosely coupled applications using dependency injection.



The Spring Project

Dependency injection is a type of inversion of control that allows us to configure the Spring container to take care of the **has-a** composition relationships for us in a flexible way (as opposed to hardcoding these relationships).



Components and Component Scan

Spring Components

01

Spring **components** are classes that Spring auto-detects and configures. We'll be using a variety of components throughout the course.

02

Spring can detect our components via component scanning. (Component scanning is default behavior for Spring Boot applications.)

Spring Component Scan can discover components (as well as many other Spring annotations) via the conjunction of classpath scanning and reflection.

Spring Component Scan

01

In a component's simplest form, all we have to do is annotate a class with `@Component`. However, we won't be writing many simple components like these. Most of our components will be `@RestController`, `@Service`, or `@Repository`. Each one of these is just a much more specific type of a component.

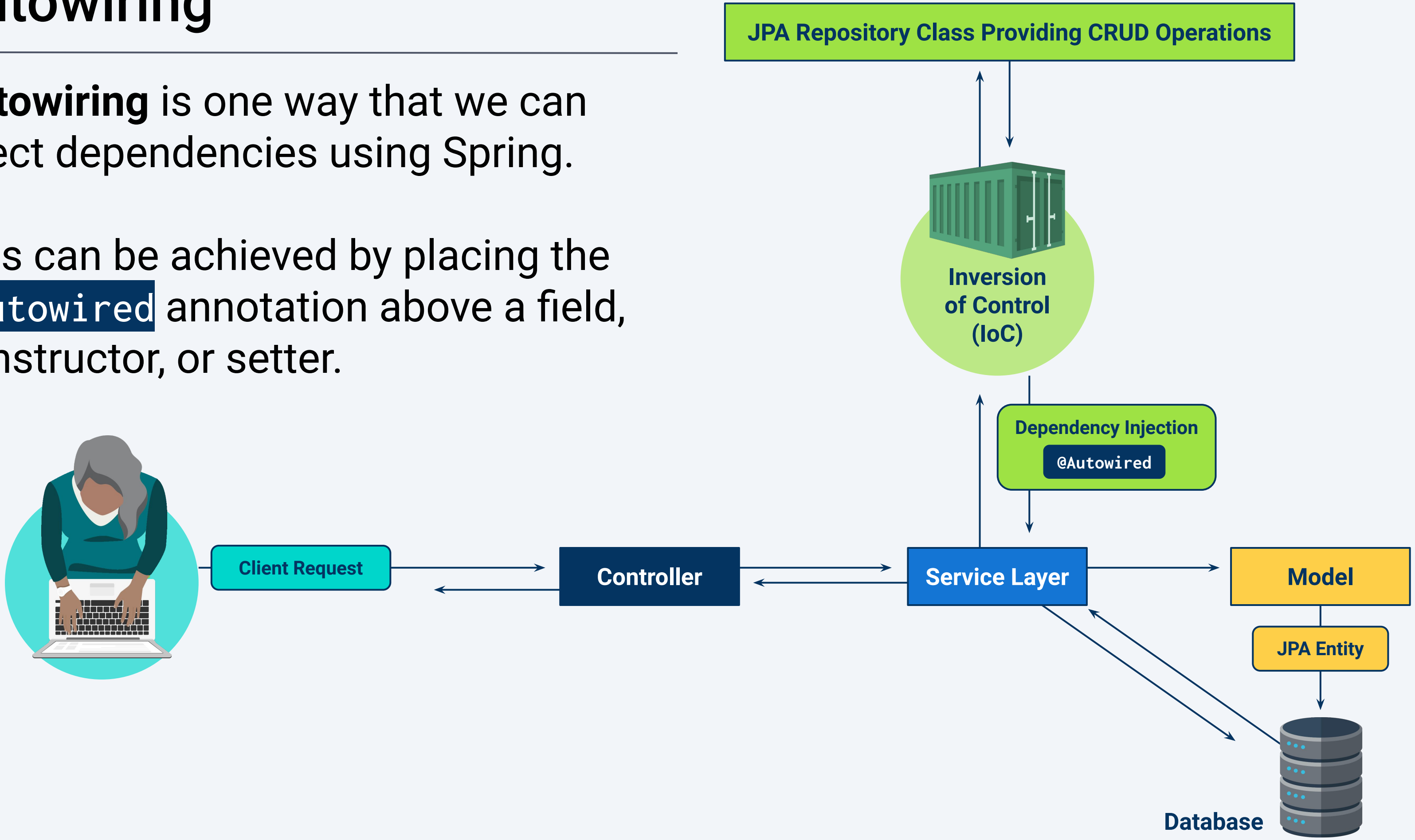
02

Whenever one of these components is needed, an appropriate Spring Bean will be created. (Spring Beans are objects that are managed by the Spring IoC container.)

Autowiring

Autowiring is one way that we can inject dependencies using Spring.

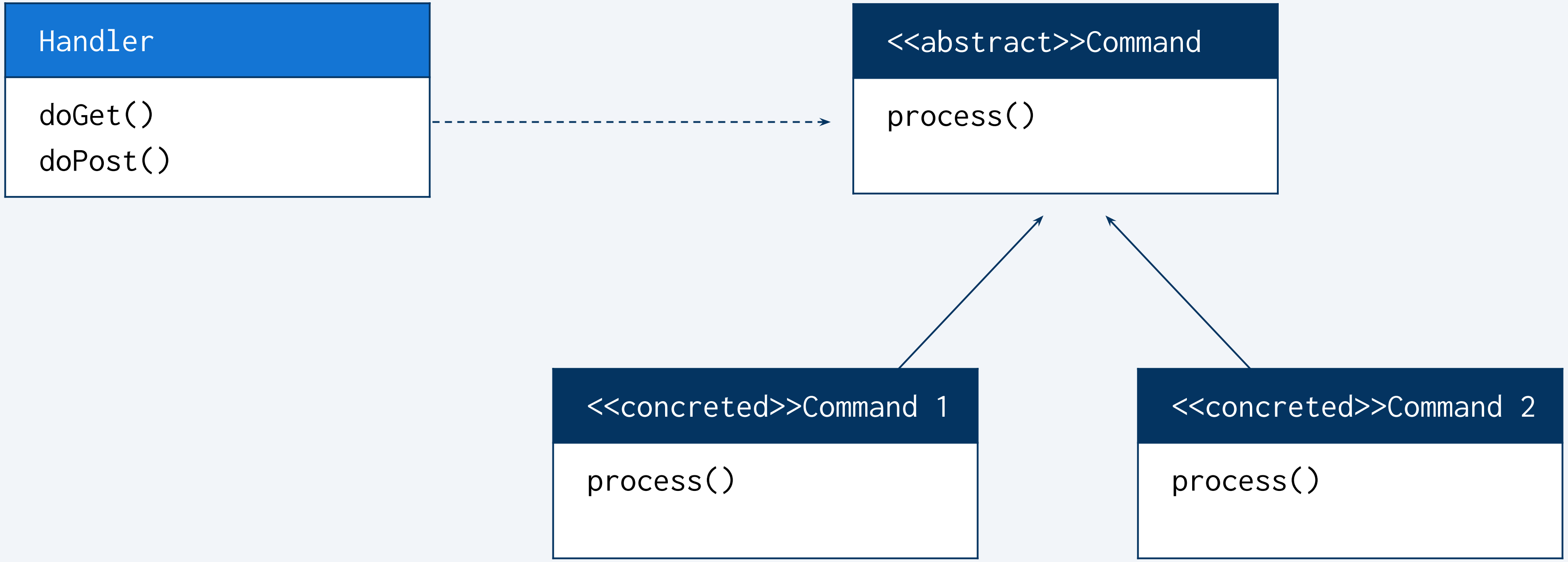
This can be achieved by placing the `@Autowired` annotation above a field, constructor, or setter.



If there exists a component whose type is compatible with the autowired variable’s type declaration, then Spring will assign the associated bean to this variable.

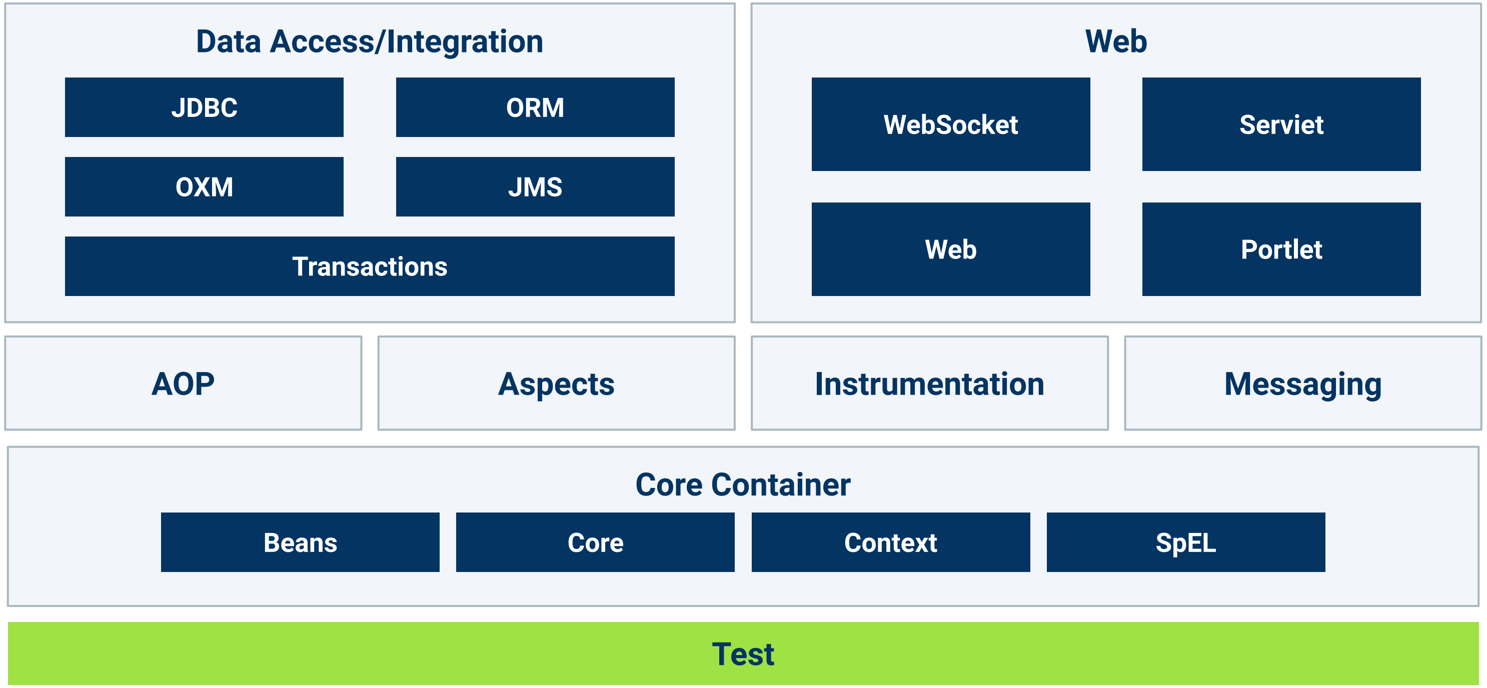
Spring MVC

Spring MVC is an MVC framework based on the **front controller** design pattern.
We'll use Spring MVC via Spring Boot.



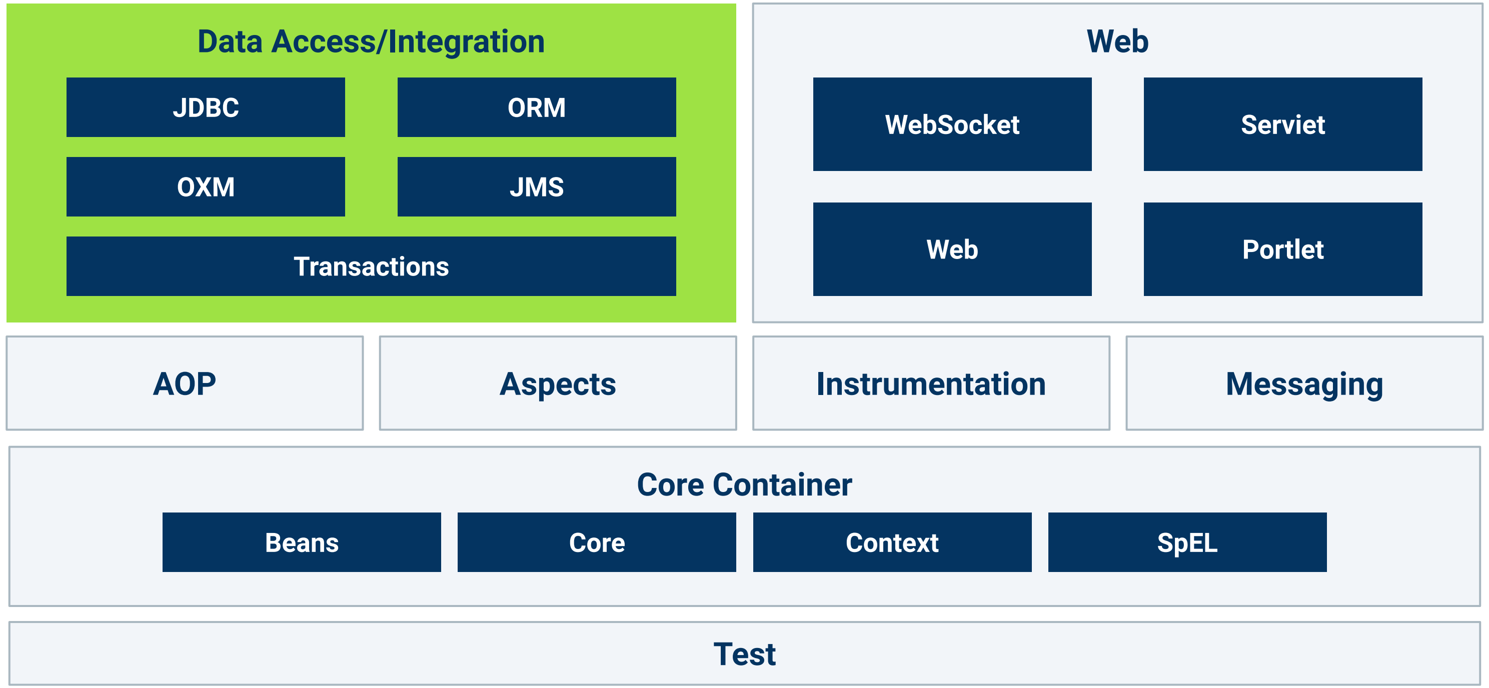
Spring MVC Testing

The Spring Framework provides testing tools such as mock objects and Spring MVC testing utilities that allow us to unit test our controllers.



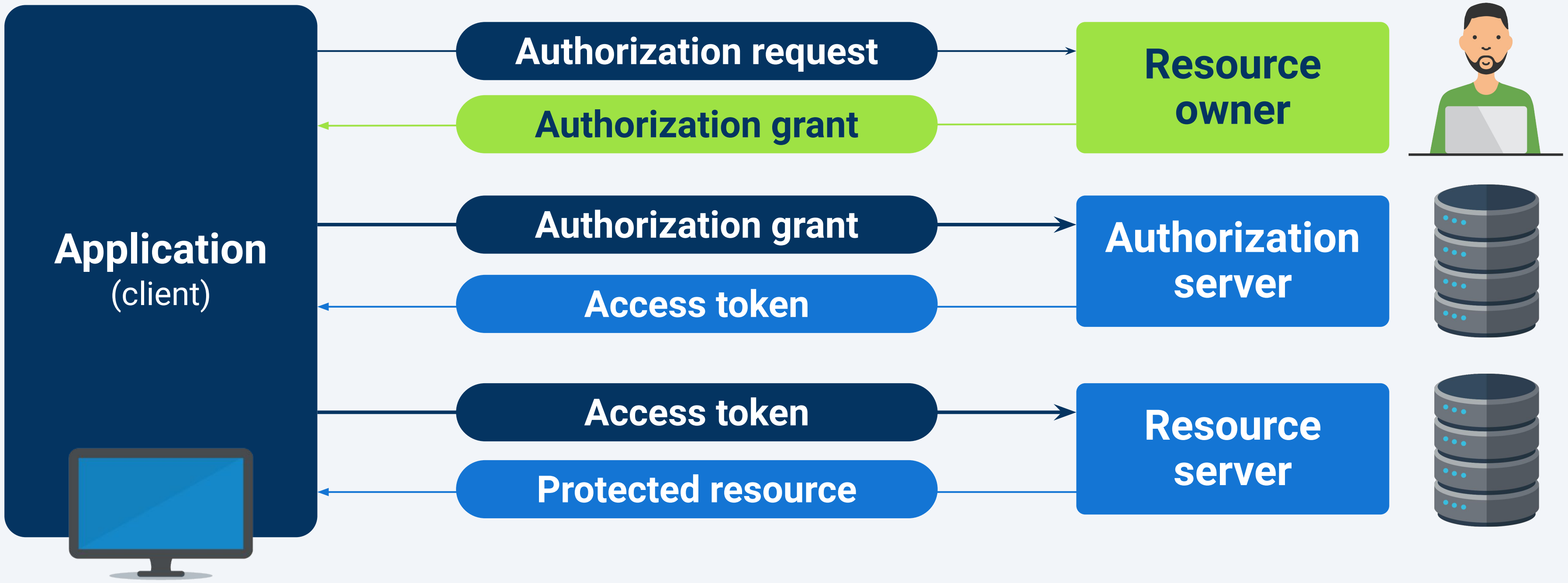
Spring Data

The Spring Data framework provides a consistent Spring-based programming model for a variety of underlying data stores. We'll be using Spring Data JPA.



Spring Security

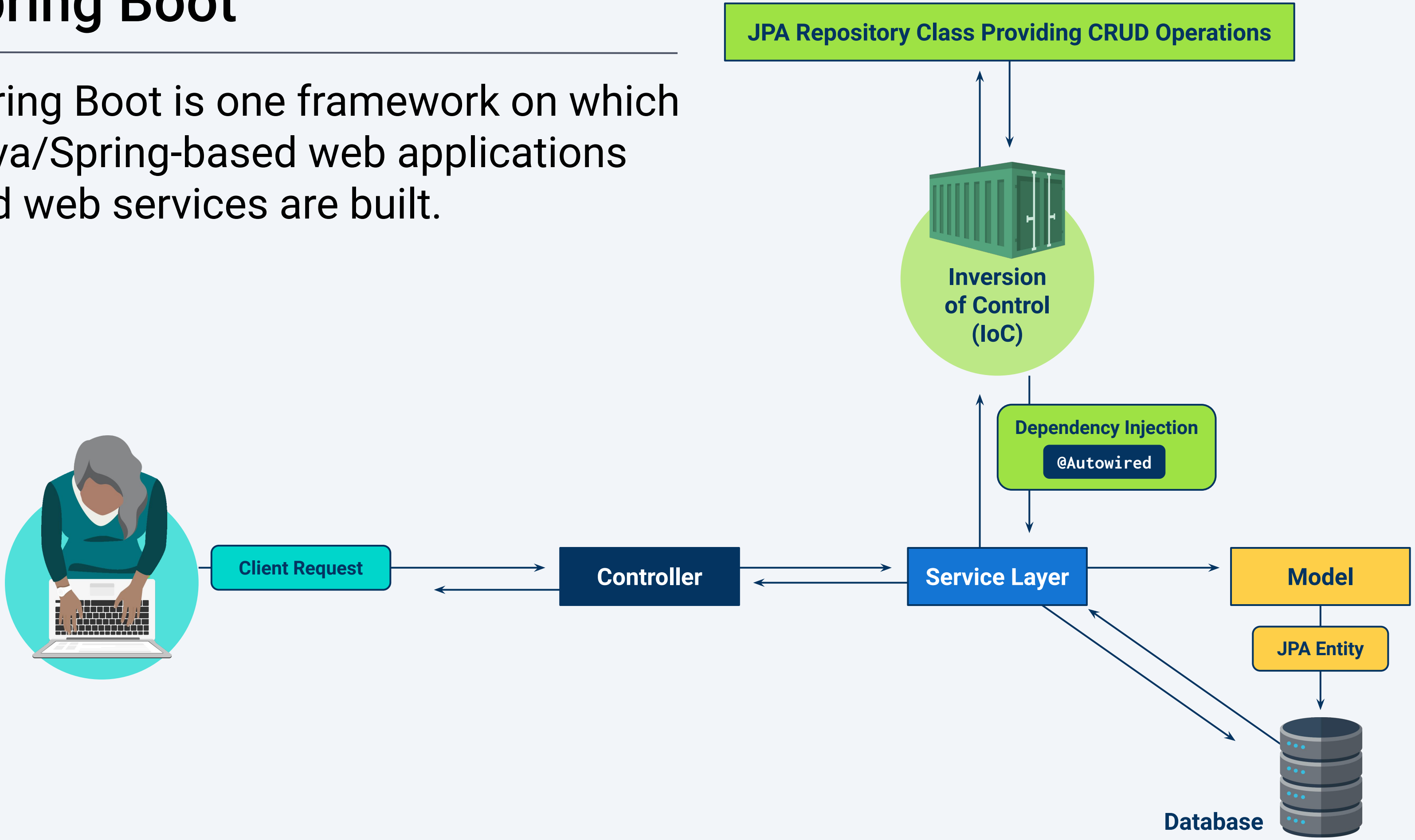
The Spring Security framework allows us to add authentication and authorization to web applications and web services.



Introduction to Spring Boot

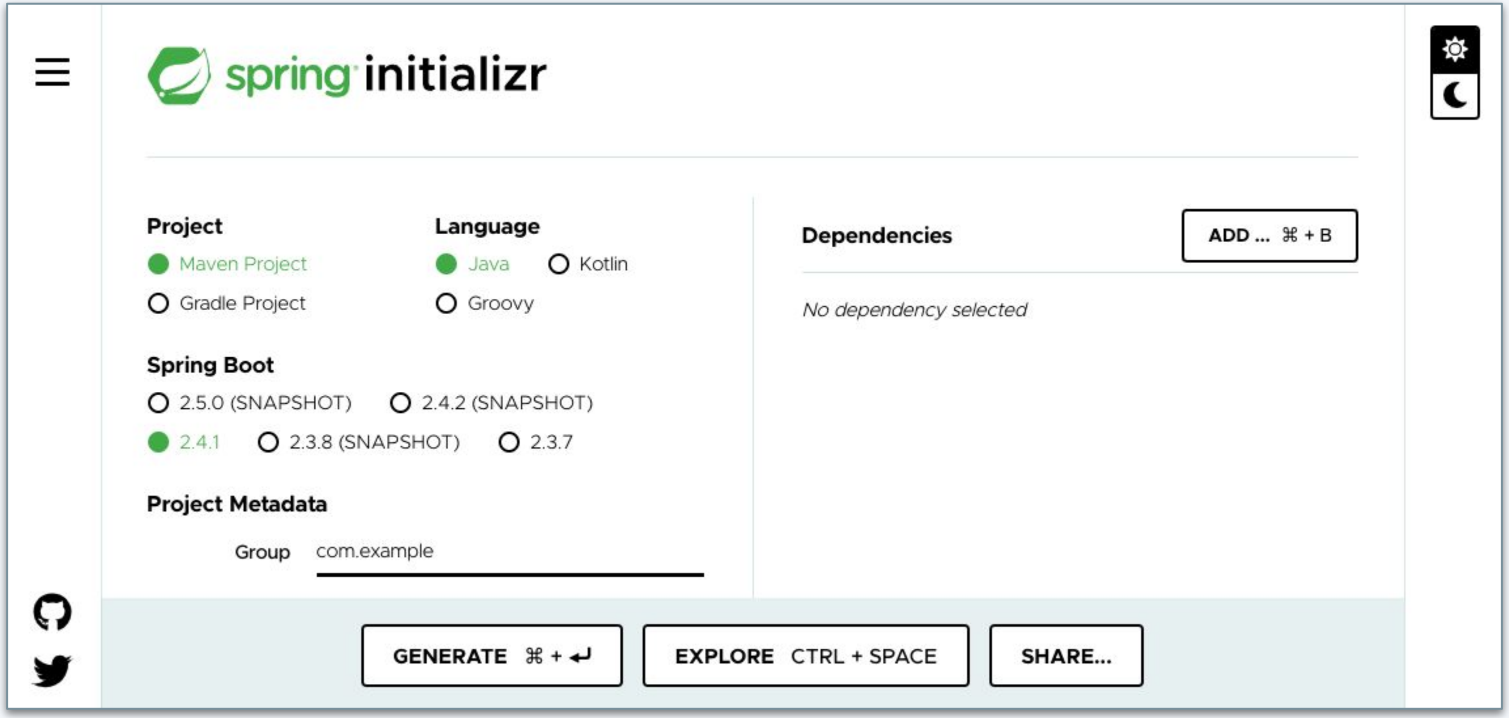
Spring Boot

Spring Boot is one framework on which Java/Spring-based web applications and web services are built.



Spring Initializr

Spring Initializr is a tool that helps us create projects with our required libraries and frameworks (such as web support), which makes starting projects much easier.


spring initializr

Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
2.5.0 (SNAPSHOT) 2.4.2 (SNAPSHOT) [✓ 2.4.1] 2.3.8 (SNAPSHOT) 2.3.7

Project Metadata
Group: com.example

Dependencies
No dependency selected

Configuration Options

- **Build system:** Maven or Gradle (we'll use Maven)
- Java version
- Spring Boot version
- Artifact and package names
- **Starter dependencies:** you tell Initializr what you want to do, and it will provide a tested and curated set of dependencies for you. This makes developing so much easier!

Output

Output results in a zip file.

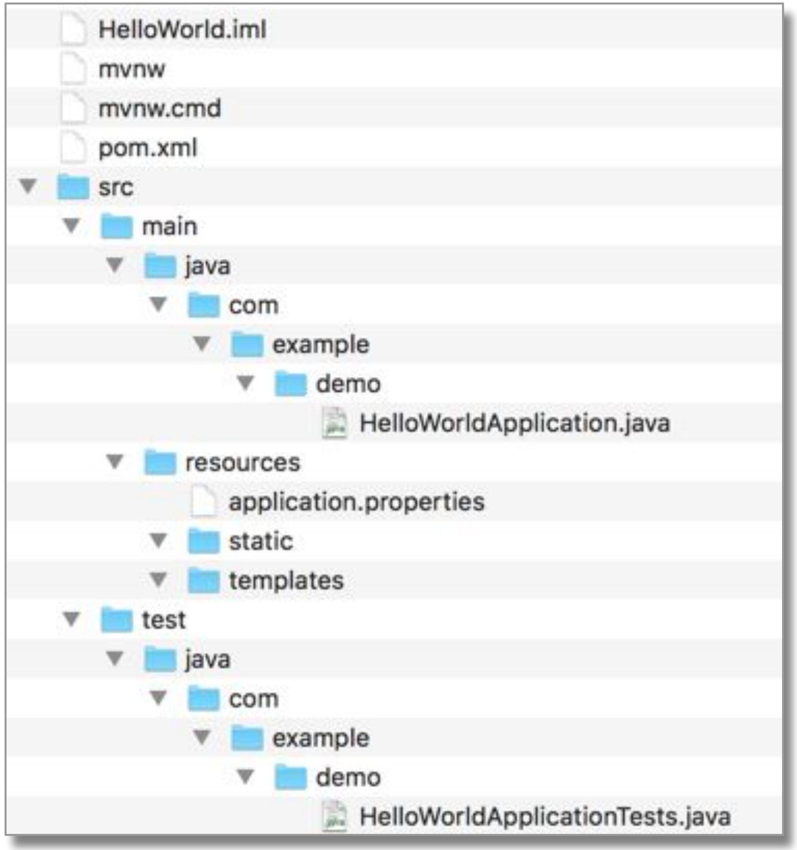
You can specify whether or not you want a traditional WAR file or an executable JAR (with an embedded server) in the advanced settings. (We'll choose JAR.)

Spring Boot Project Structure

Spring Boot projects are just
Maven projects.

There is no code generation or other hidden magic.

The curated starter dependencies are the big deal!



```
    HelloWorld.iml
    mvnw
    mvnw.cmd
    pom.xml
▼ src
    ▼ main
        ▼ java
            ▼ com
                ▼ example
                    ▼ demo
                        HelloWorldApplication.java
        ▼ resources
            application.properties
            ▼ static
            ▼ templates
    ▼ test
        ▼ java
            ▼ com
                ▼ example
                    ▼ demo
                        HelloWorldApplicationTests.java
```

Configuration Options

Spring Boot's strength is its reasonable default configuration values and auto-configuration, based on which starter dependencies and other dependencies are in the Maven POM.

Auto-Configuration: It automatically configures things based on what is on your classpath (determined by which starter and other dependencies you include in your Maven POM file).

Additional configuration can be made via:

- Annotations
- Properties files
- Java classes
- XML files (less common these days)

Time to Code

Hello, Spring Boot

Suggested Time:

Spring Annotations

@RestController

- A class-level annotation.
- 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

- A method-level annotation.

- Maps an endpoint to a method that will handle requests to that endpoint.

It has two parameters:

value	contains the URI path for this mapping.
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

- A method-level annotation.
- Indicates the HTTP status code that is sent back when the method successfully handles the incoming request.

`@PathVariable`

- A method-parameter-level annotation.
- Maps a path variable (marked by `{}` in the path) to a method parameter.
- The Spring Framework takes care of correctly converting 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 name of the method variable, no further configuration is needed.
- If they don't match, you can specify which path variable to map to the parameter like this: `@PathVariable("pathVarName")`

@RequestBody

- A method-parameter-level annotation.
- Maps JSON in the request body to a method parameter.
- A component called the Jackson Mapper takes care of converting JSON to Java and Java to JSON.

Time to Code

Student Service

Suggested Time:

Introduction to Swagger

Swagger

- Swagger Specification started in 2010. It was known as Swagger 2.0 specification.
- SmartBear Software acquired the spec and tools in 2015.
- In January 2016, the spec was transferred to a newly created organization called the OpenAPI Initiative under the Linux Foundation. It is currently known as the **OpenApi 3.0 specification**.
- Microsoft, IBM, Google, and other companies are members.

Swagger Editor

OpenAPI 3.x

The OpenAPI spec documents REST APIs.

We can document all aspects of a REST API:

- URI
- HTTP method
- Parameters
- Responses

Open API Sections

- Metadata
- Servers
- Paths

The following sections are all nested under the Paths section:

- Parameters
- Responses
- Request body
- Data definitions and refs

Time to <code>